Large-scale Clustering using Random Sketching and Validation

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Panagiotis Traganitis

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Professor Georgios B. Giannakis, Advisor

August, 2015

Acknowledgments

Firstly, I would like to extend my sincerest thanks and deepest gratitude to my advisor, Prof. Georgios B. Giannakis, for giving me the opportunity and welcoming me as his student in his prestigious research group and introducing me to the world of academic research. His vast knowledge, guidance and invaluable advice, have helped shape both this thesis and myself as an aspiring researcher.

I would also like to deeply thank Prof. Konstantinos Slavakis, whose contributions were crucial to the completion of this thesis. Special thanks go to my thesis committee members, Prof. Nikolaos Sidiropoulos and Prof. Yousef Saad. Our discussions with Prof. Sidiropoulos have always been fruitful and his advice has been pivotal, and Prof. Saad introduced me through his thorough course to the world of numerical linear algebra and its applications in my research areas.

In addition, I am grateful to all of my professors at the University of Minnesota, whose graduate courses were enlightening and gave me the necessary tools for my research. Furthermore, I would like to thank all the current and previous members of the SPiNCOM group for the fruitful discussions we have had, as well as their company.

I am very grateful to my family and close friends, without whose support and understanding I would not have made it this far.

Panagiotis Traganitis, Minneapolis, July 10, 2015

The work in this Thesis was supported by NSF grants 1343860, 1442686, 1500713, the AFOSR grant MURI-FA9550-10-1-0567, and by the European Union (European Social Fund and Greek national funds) through the Operational Program Education and Lifelong Learning of the National Strategic Reference Framework-Research Funding Program: Thalis-UoA-Secure Wireless Nonlinear Communications at the Physical Layer.

Abstract

The advent of high-speed Internet, modern devices and global connectivity has introduced the world to massive amounts of data, that are being generated, communicated and processed daily. Extracting meaningful information from this humongous volume of data is becoming increasingly challenging even for high-performance and cloud computing platforms. While critically important in a gamut of applications, clustering is computationally expensive when tasked with high-volume high-dimensional data. To render such a critical task affordable for data-intensive settings, this thesis introduces a clustering framework, named random sketching and validation (SkeVa). This framework builds upon and markedly broadens the scope of random sample and consensus RANSAC ideas that have been used successfully for robust regression. Four main algorithms are introduced, which enable clustering of high-dimensional data, as well as subspace clustering for data generated by unions of subspaces and clustering of large-scale networks. Extensive numerical tests compare the SkeVa algorithms to their state-of-the-art counterparts and showcase the potential of the SkeVa frameworks.

Contents

List of Figures

Chapter 1

Introduction

As huge amounts of data are collected perpetually from communication, imaging, and mobile devices, medical and e-commerce platforms as well as social-networking sites, this is undoubtedly an era of data deluge [1]. Such "big data" however, come with "big challenges." The sheer volume and dimensionality of data make it often impossible to run analytics and traditional inference methods using stand-alone processors, e.g., [2, 3]. In addition, as the cost of cloud computing is rapidly declining [4], there is a need for redesigning those traditional approaches to take advantage of the flexibility that has emerged from distributing required computations to multiple nodes, as well as reducing the per-node computational burden. Consequently, "workhorse" learning tools have to be re-examined in the face of today's *high-cardinality* sets possibly comprising *high-dimensional* data.

Clustering (a.k.a. unsupervised classification) refers to categorizing into groups unlabeled data encountered in the widespread applications of mining, compression, and learning tasks [5]. Among numerous clustering algorithms, K-means is the most prominent one thanks to its simplicity [5]. It thrives on "tight" groups of feature vectors, data points or objects that can be separated via (hyper)planes (Fig. 1.1). Its scope is further broadened by the so-termed probabilistic and kernel K-means, with an instantiation of the latter being equivalent to spectral clustering – the popular tool for graph-partitioning that can cope even with nonlinearly separable data [6] (Fig. 1.3). These tools can also be used to cluster non-vectorial objects as they rely only on similarities between datapoints. In addi-

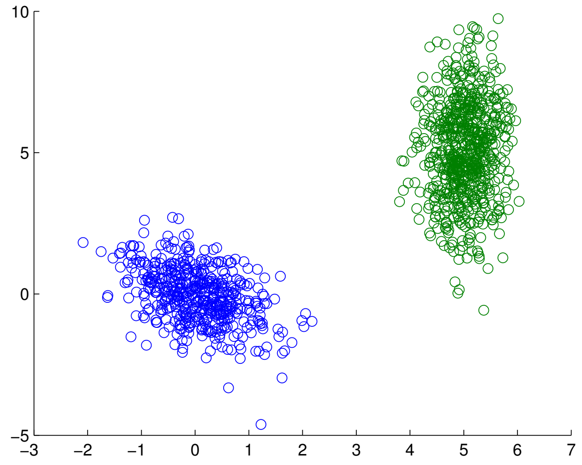


Figure 1.1: Example of a dataset with $K = 2$ clusters that are linearly separable.

tion, subspace clustering (SC) is a popular method that can group non-linearly separable data by assuming they are generated by a union of (affine) subspaces in a high-dimensional Euclidean space [7]. SC enjoys wide applicability, with applications that span image and video segmentation to identification of switched linear systems in control theory [7].

Furthermore, clustering algorithms can also find applications in graphs and networks (Figs. 1.4,1.5). As nowadays large-scale networks generate massive amounts of data, numerous challenges are posed regarding their efficient processing [8]. Many of these networks, such as social and biological, exhibit community structure which is indicative of groups (clusters) within which the edge density is relatively high compared to the edge density between groups [9]. These communities generally denote common roles or similar behavior between nodes of the network, e.g., "friends" on Facebook or "followers" on Twitter. The task of identifying these communities is similar to clustering and has received a lot of attention recently, especially in the context of social networks [10].

1.1 Context and motivation

A key question with regards to clustering data sets of cardinality N containing D-dimensional vectors with N and/or D huge, is: How can one select the most "informative"

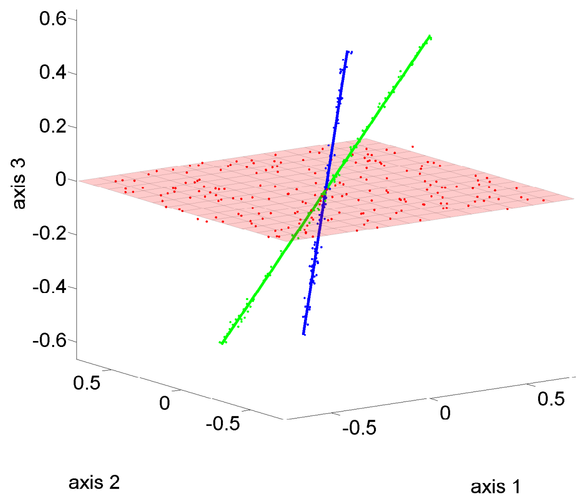


Figure 1.2: Example of dataset generated by the union of $K = 3$ subspaces.

vectors and/or dimensions so as to reduce their number for efficient computations, yet retain as much of their cluster-discrimination capability?

Feature selection is a rich topic [11] explored extensively from various angles, including pattern recognition, source coding and information theory, (combinatorial) optimization [12, 13], and neural networks [14]. Unfortunately, most available selection schemes do not scale well with the number of features D, particularly in the big data regime where D is massive. Recent approaches to dimensionality reduction and clustering include subspace clustering, where minimization problems requiring singular value decompositions (SVDs) are solved per iteration to determine in parallel a low-dimensional latent space and corresponding sparse coefficients for efficient clustering [15]. Low-dimensional subspace representations are also pursued in the context of kernel K-means [16, Alg. 2], where either an SVD on a sub-sampled kernel matrix, or, the solution of a quadratic optimization task is required per iteration to cluster efficiently large-scale data.

Randomized schemes select features with non-uniform probabilities that depend on the so-termed "leverage scores" of the data matrix [17, 18]. Unfortunately, their complexity is loaded by the leverage scores computation, which, similar to [15], requires SVD com-

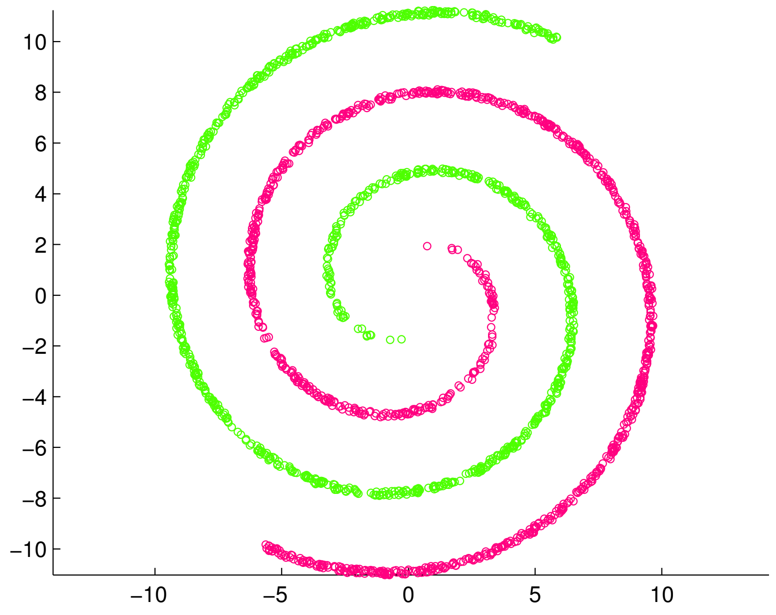


Figure 1.3: Example of a dataset with $K = 2$ clusters that are not linearly separable.

putations – a cumbersome (if not impossible) task when $D \gg$ and/or $N \gg$. Recent computationally efficient alternatives for feature selection and clustering rely on random projections (RPs) [17–20]. Specifically for RP-based clustering [20], the data matrix is left multiplied by a data-agnostic (fat) $d \times D$ RP matrix to reduce its dimension ($d \ll D$); see also [21] where RPs are employed to accelerate the kernel K-means algorithm. Clustering is performed afterwards on the reduced d-dimensional vectors. With its universality and quantifiable performance granted, this "one-RP-matrix-fits-all" approach is not as flexible to adapt to the data-specific attributes (e.g., low rank) that is typically present in big data.

Multiple algorithms have been developed by the machine learning [7] and data mining [22] communities for the task of subspace clustering. The more recent SC methods can offer high levels of clustering performance, at the cost, however, of high computational complexity. The main computational burden in these algorithms is the large number of datapoints, which have a quadratic effect on the required complexity. A key question in this case, is whether high clustering performance can be maintained while not exceeding

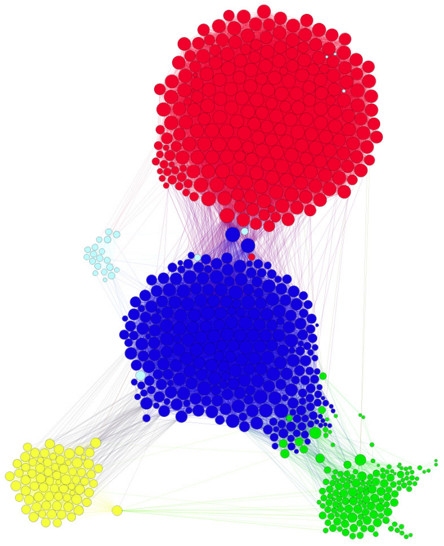


Figure 1.4: Facebook egonet with $N = 744$ vertices, $|\mathcal{E}| = 30,023$ edges, and $K = 5$ communities.

the computational budget that is available.

In addition, a plethora of algorithms have been developed for community identification purposes. Since networks can be considered as graphs, most of the community identification algorithms are related to graph partitioning or aim at optimizing graph related metrics, such as modularity [23]. However, as the number of communities and their sizes increase, there is an urgent need for scalable algorithms able to handle the immense amount of data. For this purpose, clustering, can be employed to identify communities in networks where grouping data has strong ties with community identification. Specifically, spectral clustering [24], one of the "workhorse" algorithms in clustering, shows great potential for community identification as it presumes a graph structure on the input data, and can be consequently used as an "off-the-shelf" tool for clustering communities in networks. However, since spectral clustering entails eigenvalue decompositions and realizations of K-means, the computational complexity may become prohibitively high in large-scale networks.

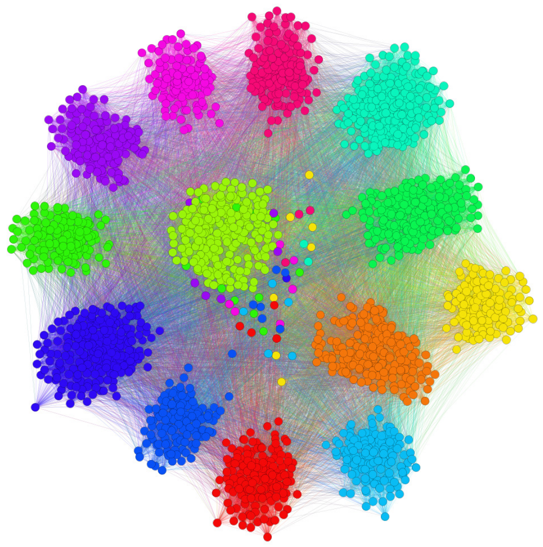


Figure 1.5: Synthetic network with $N = 10,000$ vertices, $|\mathcal{E}| = 250,214$ edges, and $K = 13$ communities.

1.2 Thesis contributions

This thesis aims to provide a framework for clustering large-scale datasets and networks, using randomization. Albeit distinct, the inspiration comes from random sampling and consensus (RANSAC) methods, which were introduced for outlier-resilient regression tasks encountered in computer vision [25–30].

This thesis's approach aspires to not only account for structure, but also offer a gamut of novel randomized algorithms trading off complexity for clustering performance by developing a family of what is termed sketching and validation (SkeVa) algorithms. The SkeVa family includes two algorithms based on efficient intermediate K-means clustering steps. The first is a batch method, while the second is sequential thus offering computational efficiency and suitability for streaming modes of operation.

The third member of the SkeVa family is kernel-based and enables big data clustering of even nonlinearly separable data sets. In addition, this method can be employed to cluster large-scale networks by exploiting the relation of kernel methods to spectral clustering.

Finally, the fourth member of the SkeVa family bypasses the need for intermediate K-means clustering thus trading off performance for complexity. This method borrows ideas from kernel-based approximation of probability density functions (pdfs) [31], and a variant of it can be used for subspace clustering of high-volume datasets.

Most of the proposed algorithms have the potential of massive parallelization, and therefore can be used also in a distributed fashion. This enables using them in a massive datacenter or cloud computing platform. Extensive numerical validations on synthetic and real data-sets highlight the potential of the proposed methods, and demonstrate their competitive performance on clustering massive populations of high-dimensional data versus state-of-the-art alternatives.

1.3 Thesis outline

The remainder of this thesis is organized as follows.

- Chapter 2 introduces the family of SkeVa algorithms for Big Data clustering, along with extensive numerical tests.
- In Chapter 3 an extension of SkeVa is introduced for subspace clustering along with numerical tests and a rigorous performance analysis.
- Chapter 4 presents a SkeVa algorithm for identification of communities in large-scale networks.
- Chapter 5 presents a concluding discussion on the SkeVa approaches, as well as a brief discussion on future research thrusts.

1.4 Notational conventions

Lowercase bold letters, $\boldsymbol{x}$, denote vectors, uppercase bold letters, $\mathbf{X}$, denote matrices, and calligraphic uppercase letters, $\mathcal{X}$, denote sets. The (i, j)th entry of matrix $\mathbf{X}$ is denoted by $[\mathbf{X}]_{ij}$. Moreover, $\mathbb{R}^D$ stands for the D-dimensional real Euclidean space, $\mathbb{E}[\cdot]$ the expectation of a random variable, and $\|\cdot\|$ a norm.

Chapter 2

Big Data Clustering via Random Sketching and Validation

In response to the need for learning tools tuned to big data analytics, the present chapter introduces a framework for efficient clustering of huge sets of (possibly high-dimensional) data. Building on random sampling and consensus (RANSAC) ideas pursued earlier in a different (computer vision) context for robust regression, a suite of novel dimensionality- and set-reduction algorithms is developed. The advocated *sketch-and-validate* (SkeVa) family includes two algorithms that rely on K-means clustering per iteration on reduced number of dimensions and/or feature vectors: The first operates in a batch fashion, while the second sequential one offers computational efficiency and suitability with streaming modes of operation. For clustering even nonlinearly separable vectors, the SkeVa family offers also a member based on user-selected kernel functions. Further trading off performance for reduced complexity, a fourth member of the SkeVa family is based on a divergence criterion for selecting proper minimal subsets of feature variables and vectors, thus bypassing the need for K-means clustering per iteration. Extensive numerical tests on synthetic and real data sets highlight the potential of the proposed algorithms, and demonstrate their competitive performance relative to state-of-the-art random projection alternatives.

2.1 Preliminaries

Consider the $D \times N$ data matrix $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ with D and/or N being potentially massive. Data $\{\boldsymbol{x}_n\}_{n=1}^N$ belong to a known number of K clusters ($K \ll N$). Each cluster $\mathcal{C}_k$ is represented by its centroid $\boldsymbol{c}_k$ that can be e.g., the (sample) mean of the vectors in $\mathcal{C}_k$. Accordingly, each datum can be modeled as $\boldsymbol{x}_n = \boldsymbol{C}\boldsymbol{\pi}_n + \boldsymbol{v}_n$, where $\boldsymbol{C} := [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K]$; the sparse $K \times 1$ vector $\boldsymbol{\pi}_n$ comprises the data-cluster association entries satisfying $\sum_{k=1}^K [\boldsymbol{\pi}_n]_k = 1$, where $[\boldsymbol{\pi}_n]_k \in (0,1]$ when $\boldsymbol{x}_n \in \mathcal{C}_k$, while $[\boldsymbol{\pi}_n]_k = 0$, otherwise; and the noise $\boldsymbol{v}_n$ captures $\boldsymbol{x}_n$'s deviation from the corresponding centroid(s).

For *hard clustering*, the said associations are binary ($[\boldsymbol{\pi}_n]_k \in \{0,1\}$), and in the celebrated hard K-means algorithm they are identified based on the Euclidean (ℓ_2) distance between $\boldsymbol{x}_n$ and its closest centroid. Specifically, given K and $\{\boldsymbol{x}_n\}_{n=1}^N$, per iteration $i = 1, 2, \ldots$, the K-means algorithm iteratively updates data-cluster associations and cluster centroids as follows; see e.g., [5].

[i-a] **Update data-cluster associations:** For $n = 1, \ldots, N$,

$$\boldsymbol{x}_n \in \mathcal{C}_k[i] \Leftrightarrow k \in \arg \min_{k' \in \{1,\ldots,K\}} \|\boldsymbol{x}_n - \boldsymbol{c}_{k'}[i]\|_2^2 \; . \tag{2.1a}$$

[i-b] **Update cluster centroids:** For $k = 1, \ldots, K$,

$$\begin{aligned} \boldsymbol{c}_k[i+1] &\in \arg\min_{\boldsymbol{c} \in \mathbb{R}^D} \sum_{\boldsymbol{x}_n \in \mathcal{C}_k[i]} \|\boldsymbol{x}_n - \boldsymbol{c}\|_2^2 \\ &= \frac{1}{\left|\mathcal{C}_k[i]\right|} \sum\nolimits_{\boldsymbol{x}_n \in \mathcal{C}_k[i]} \boldsymbol{x}_n \; . \end{aligned} \tag{2.1b}$$

Although there may be multiple assignments solving (2.1a), each $\boldsymbol{x}_n$ is assigned to a single cluster. To initialize (2.1a), one can randomly pick $\{\boldsymbol{c}_k[1]\}_{k=1}^K$ from $\{\boldsymbol{x}_n\}_{n=1}^N$. The iterative algorithm (2.1) solves a challenging NP-hard problem, and albeit its success, K-means guarantees convergence only to a local minimum at complexity $\mathcal{O}(NDKI)$, with I denoting the number of iterations needed for convergence, which depends on initialization [5, § 9.1].

Remark 1. *As (2.1a) and (2.1b) minimize an ℓ_2-norm squared, hard K-means is sensitive to outliers. Variants exhibiting robustness to outliers adopt non-Euclidean distances (a.k.a.*

dissimilarity metrics) δ, such as the ℓ_1-norm. In addition, candidate "centroids" can be selected per iteration from the data themselves; that is, $\mathbf{c} \in \mathcal{C}_k[i]$ in (2.1b). Notwithstanding for this so-termed K-medoids algorithm, one just needs the distances $\{\delta(\boldsymbol{x}_n, \boldsymbol{x}_{n'})\}$ to carry out the minimizations in (2.1a) and (2.1b). The latter in turn allows $\{\boldsymbol{x}_n\}_{n=1}^N$ to even represent non-vectorial *objects (a.k.a. qualitative data), so long as the aforementioned (non-)Euclidean distances can become available otherwise; e.g., in the form of correlations [5, § 9.1].*

Besides various distances and centroid options, hard K-means in (2.1) can be generalized in three additional directions: (i) Using nonlinear functions $\varphi : \mathbb{R}^D \to \mathcal{H}$, with $\mathcal{H}$ being a potentially infinite-dimensional space, data $\{\boldsymbol{x}_n\}$ can be transformed to $\{\varphi(\boldsymbol{x}_n)\}$, where clustering can be facilitated (cf. Sec. 2.2.3); (ii) via non-binary $\boldsymbol{\pi}_n \in [0,1]^K$, *soft clustering* can allow for multiple associations per datum, and thus for a probabilistic assignment of data to clusters; and (iii) additional constraints (e.g., sparsity) can be incorporated in the $[\boldsymbol{\pi}_n]_k$ coefficients through appropriate regularizers $\rho(\boldsymbol{\pi})$.

All these generalizations can be unified by replacing (2.1) per iteration $i = 1, 2, \ldots$, with

[i-a] **Update data-cluster associations:** $n = 1, \ldots, N$,

$$\boldsymbol{\pi}_n[i] \in \arg \min_{\substack{\boldsymbol{\pi} \in [0,1]^K; \\ \mathbf{1}^\top \boldsymbol{\pi} = 1}} \delta\left(\varphi(\boldsymbol{x}_n), \sum_{k=1}^K \pi_k \boldsymbol{c}_k[i]\right) + \rho(\boldsymbol{\pi})\,. \tag{2.2a}$$

[i-b] **Update cluster centroids:**

$$\{\boldsymbol{c}_k[i+1]\}_{k=1}^K \in \arg \min_{\{\boldsymbol{c}_k\}_{k=1}^K \subset \mathcal{H}} \sum_{n=1}^N \delta\left(\varphi(\boldsymbol{x}_n), \sum_{k=1}^K \big[\boldsymbol{\pi}_n[i]\big]_k \boldsymbol{c}_k\right)\,. \tag{2.2b}$$

In Sec. 2.2.3, function φ will be implicitly used to map nonlinearly separable data $\{\boldsymbol{x}_n\}_{n=1}^N$ to linearly separable (possibly infinite dimensional) data $\{\varphi(\boldsymbol{x}_n)\}_{n=1}^N$, whose distances can be obtained through a pre-selected (so-termed kernel) function κ [5, Chap. 6]. The regularizer $\rho(\boldsymbol{\pi})$ can enforce prior knowledge on the data-cluster association vectors.

To confirm that indeed (2.1) is subsumed by (2.2), let $\varphi(\boldsymbol{x}_n) = \boldsymbol{x}_n$; choose δ as the squared Euclidean distance in $\mathbb{R}^D$; and set $\rho(\boldsymbol{\pi}) = 0$, if $\boldsymbol{\pi} \in \mathcal{E}_K := \{\boldsymbol{e}_k\}_{k=1}^K$, whereas

$\rho(\boldsymbol{\pi}) = +\infty$ otherwise, with $\mathbf{e}_k$ being the kth K-dimensional canonical vector. Then (2.2a) becomes $\boldsymbol{\pi}_n[i] \in \arg\min_{\boldsymbol{\pi}\in\{0,1\}^K;\,\mathbf{1}^\top\boldsymbol{\pi}=1} \|\boldsymbol{x}_n - \sum_k \pi_k \boldsymbol{c}_k[i]\|_2^2$, which further simplifies to the minimum distance rule of (2.1a). Moreover, it can be readily verified that (2.2b) $\{\boldsymbol{c}_k[i+1]\}_{k=1}^K \in \arg\min_{\{\boldsymbol{c}_k\}_{k=1}^K} \sum_n \|\boldsymbol{x}_n - \sum_k [\boldsymbol{\pi}_n[i]]_k \boldsymbol{c}_k\|_2^2$ separates across ks to yield the centroid of (2.1b) per cluster.

To recognize how (2.2) captures also soft clustering, consider that cluster $\mathcal{C}_k$ is selected with probability $\pi_k := \Pr(\mathcal{C}_k)$, and its data are drawn from a probability density function (pdf) p parameterized by $\boldsymbol{\theta}_k$; i.e., $\boldsymbol{x}|\mathcal{C}_k \sim p(\boldsymbol{x};\boldsymbol{\theta}_k)$. If p is Gaussian, then $\boldsymbol{\theta}_k$ denotes its mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. With $\boldsymbol{\theta} := [\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_K^\top]^\top$ and allowing for multiple cluster associations, the likelihood per datum is given by the mixture pdf: $p(\boldsymbol{x};\boldsymbol{\pi},\boldsymbol{\theta}) = \sum_{k=1}^K \pi_k p(\boldsymbol{x};\boldsymbol{\theta}_k)$, which for independently drawn data yields the joint log-likelihood ($\boldsymbol{\Pi} := [\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N]$)

$$\ln p(\boldsymbol{X};\boldsymbol{\Pi},\boldsymbol{\theta}) = \sum_{n=1}^N \ln\left(\sum_{k=1}^K [\boldsymbol{\pi}_n]_k p(\boldsymbol{x}_n;\boldsymbol{\theta}_k)\right). \tag{2.3}$$

If $\varphi(\boldsymbol{x}_n) := \boldsymbol{x}_n$, $\boldsymbol{c}_k := p(\boldsymbol{x}_n;\boldsymbol{\theta}_k)$, and $\delta(\boldsymbol{x}_n, \sum_k \pi_k c_k) := -\ln(\sum_k [\boldsymbol{\pi}_n]_k p(\boldsymbol{x}_n;\boldsymbol{\theta}_k))$ in (2.2), then soft K-means iterations (2.2a) and (2.2b) maximize (2.3) with respect to (w.r.t.) $\boldsymbol{\Pi}$ and $\boldsymbol{\theta}$. An alternative popular maximizer of the likelihood in (2.3) is the *expectation-maximization* algorithm; see e.g., [5, § 9.3].

If the number of clusters K is unknown, it can also be estimated by regularizing the log-likelihood in (2.3) with terms penalizing complexity as in e.g., minimum description length criteria [5].

Although hard K-means is the clustering module common to all numerical tests in Sec. V, the novel big data algorithms of Secs. III and IV apply to all schemes subsumed by (2.2).

2.2 The SkeVa Family

Five novel algorithms based on random sketching and validation are introduced in this section. Relative to existing clustering schemes, their merits are pronounced when D and/or N take prohibitively large values for the unified iterations (2.2a) and (2.2b) to remain computationally feasible.

2.2.1 Batch algorithm

For specificity, the SkeVa K-means algorithm will be developed first for $D \gg$, followed by its variant for $N \gg$.

Using a repeated trial-and-error approach, SkeVa K-means discovers a few dimensions (features) that yield high-accuracy clustering. The key idea is that upon sketching a small enough subset of dimensions (trial or sketching phase), a hypotheses test can be formed by augmenting the original subset with a second small subset (up to d affordable dimensions) to validate whether the first subset nominally represents the full D-dimensional data (error phase). Such a trial-and-error procedure is repeated for a number $R_{\max}$ of realizations, after which the features that have achieved the "best" clustering accuracy results are those determining the final clusters on the whole set of dimensions.

Starting with the trial-phase per realization r, $\check{d}$ dimensions (rows) of $\boldsymbol{X}$ are randomly drawn (uniformly) to obtain $\check{\boldsymbol{X}}^{(r)} := [\check{\boldsymbol{x}}_1^{(r)}, \ldots, \check{\boldsymbol{x}}_N^{(r)}] \in \mathbb{R}^{\check{d} \times N}$. With $\check{d}$ small enough, K-means is run on $\check{\boldsymbol{X}}^{(r)}$ to obtain clusters $\{\check{\mathcal{C}}_k^{(r)}\}_{k=1}^K$ and corresponding centroids $\{\check{\boldsymbol{c}}_k^{(r)}\}_{k=1}^K$ [cf. (2.1a) and (2.1b)]. These sketching and clustering steps comprise the *(random) sketching* phase.

Moving on to the error-phase of the procedure, the quality of the $\check{d}$-dimensional clustering is assessed next using what is termed *validation* phase. This starts by re-drawing $\check{d}'$-dimensional data $\{\check{\boldsymbol{x}}_n^{(r')}\}_{n=1}^N$ ($\check{d}' \ll D - \check{d}$), generally different from those selected in draw r. Associating each $\check{\boldsymbol{x}}_n^{(r')}$ with the cluster $\check{\boldsymbol{x}}_n^{(r)}$ belongs to, the centroids corresponding to the extra $\check{d}'$ dimensions are formed as [cf. (2.1b)]

$$\check{\boldsymbol{c}}_k^{(r')} = \frac{1}{|\check{\mathcal{C}}_k^{(r)}|} \sum_{\check{\boldsymbol{x}}_n^{(r)} \in \check{\mathcal{C}}_k^{(r)}} \check{\boldsymbol{x}}_n^{(r')} . \tag{2.4}$$

Let $\bar{\boldsymbol{x}}_n^{(r)} := [\check{\boldsymbol{x}}_n^{(r)\top}, \check{\boldsymbol{x}}_n^{(r')\top}]^\top$ and $\bar{\boldsymbol{c}}_k^{(r)} := [\check{\boldsymbol{c}}_k^{(r)\top}, \check{\boldsymbol{c}}_k^{(r')\top}]^\top$ denote respectively the concatenated data and centroids from draws r and r', and likewise for the data and centroid matrices $\bar{\boldsymbol{X}}^{(r)}$ and $\bar{\boldsymbol{C}}^{(r)}$. Measuring distances $\{\delta(\bar{\boldsymbol{x}}_n^{(r)}, \bar{\boldsymbol{c}}_k^{(r)})\}$ and using again the minimum distance rule data-cluster associations and clusters $\{\bar{\mathcal{C}}_k^{(r)}\}_{k=1}^K$ are obtained for the "augmented data."

If per datum $\boldsymbol{x}_n$ the data-cluster association in the space of $\check{d}$ dimensions coincides with that in the space of $d := \check{d} + \check{d}'$ dimensions, then $\boldsymbol{x}_n$ is in the *validation set* (VS) $\mathcal{V}_D^{(r)}$; that is,

$$\mathcal{V}_D^{(r)} := \left\{ \boldsymbol{x}_n \;\middle|\; \check{\boldsymbol{x}}_n^{(r)} \in \check{\mathcal{C}}_{k_1}^{(r)}, \bar{\boldsymbol{x}}_n^{(r)} \in \bar{\mathcal{C}}_{k_2}^{(r)}, \text{ and } k_1 = k_2 \right\} . \tag{2.5}$$

Quality of clustering per draw is then assessed using a monotonically increasing rank function f of the set $\mathcal{V}_D^{(r)}$. Based on this function, a $\check{d}$-dimensional trial r_1 is preferred over another $\check{d}$-dimensional trial r_2 if $f(\mathcal{V}_D^{(r_1)}) > f(\mathcal{V}_D^{(r_2)})$.

The sketching and validation phases are repeated for a prescribed number of realizations $R_{\max}$. At last, the $\check{d}$-dimensional sketching $r_* := \arg\max_{r \in \{1,\ldots,R_{\max}\}} f(\mathcal{V}_D^{(r)})$ yields the final clusters, namely $\{\check{\mathcal{C}}_k^{(r_*)}\}_{k=1}^K$; see Alg. 2.1.

Algorithm 2.1 Batch SkeVa K-means

```
Input: Data $\boldsymbol{X}$; number of clusters $K$; reduced dimensions $\check{d}$ and $\check{d}'$ for the sketching
    and validation phases, respectively; ranking function $f$; number of realizations (draws)
    $R_{\max}$.
Output: Data-cluster associations on $\boldsymbol{X}$.
 1: for $r = 1$ to $R_{\max}$ do
 2:   Randomly sample $\check{d} \ll D$ rows of $\boldsymbol{X}$ to obtain $\check{\boldsymbol{X}}^{(r)}$.
 3:   Run $K$-means on $\check{\boldsymbol{X}}^{(r)}$; obtain clusters $\{\check{\mathcal{C}}_k^{(r)}\}_{k=1}^K$ and centroids $\{\check{\boldsymbol{c}}_k^{(r)}\}_{k=1}^K$ [cf. (2.1)].
 4:   Randomly sample $\check{d}' \ll D$ rows of $\boldsymbol{X}$ (other than those in step 2) to obtain $\check{\boldsymbol{X}}^{(r')}$.
 5:   Per cluster $k$, form $\bar{\boldsymbol{c}}_k^{(r)} := [\check{\boldsymbol{c}}_k^{(r)\top}, \check{\boldsymbol{c}}_k^{(r')\top}]^\top$.
 6:   Associate $\{\bar{\boldsymbol{x}}_n^{(r)}\}_{n=1}^N$ to closest $\{\bar{\boldsymbol{c}}_k^{(r)}\}_{k=1}^K$.
 7:   Identify validation set $\mathcal{V}_D^{(r)}$ [cf. (2.5)].
 8: end for
 9: $r_* := \arg\max_{r \in \{1,\ldots,R_{\max}\}} f(\mathcal{V}_D^{(r)})$.
10: Associate data to clusters on $\boldsymbol{X}$ as in $\{\check{\mathcal{C}}_k^{(r_*)}\}_{k=1}^K$.
```

With regards to selecting f, a straightforward choice is the VS cardinality, that is $f(\mathcal{V}_D^{(r)}) := |\mathcal{V}_D^{(r)}|$, which can be thought as the empirical probability of correct clustering. Alternatively, a measure of cluster separability, used extensively in pattern recognition, is

Fisher's discriminant ratio [5], which in the present context becomes

$$\mathrm{FDR}^{(r)} := \sum_{k_1=1}^{K} \sum_{\substack{k_2=1;\\ k_2\neq k_1}}^{K} \frac{\left\|\bar{\boldsymbol{c}}_{k_1}^{(r)} - \bar{\boldsymbol{c}}_{k_2}^{(r)}\right\|_2^2}{\left(\bar{\sigma}_{k_1}^{(r)}\right)^2 + \left(\bar{\sigma}_{k_2}^{(r)}\right)^2} \tag{2.6}$$

where $(\bar{\sigma}_k^{(r)})^2$ is the unbiased sample variance of cluster k:

$$\left(\bar{\sigma}_k^{(r)}\right)^2 := \frac{1}{|\check{\mathcal{C}}_k^{(r)}| - 1} \sum_{\check{\boldsymbol{x}}_n \in \check{\mathcal{C}}_k^{(r)}} \left\|\bar{\boldsymbol{x}}_n^{(r)} - \bar{\boldsymbol{c}}_k^{(r)}\right\|_2^2 . \tag{2.7}$$

The larger the $\mathrm{FDR}^{(r)}$, the more separable clusters are. Obtaining $\mathrm{FDR}^{(r)}$ is computationally light since distances in (2.7) have been calculated during the validation phase of the algorithm. The only additional burden is computing the numerator in (2.6) in $\mathcal{O}[(\check{d}+\check{d}')K^2]$ complexity. Based on FDR, a second choice for f is

$$f(\mathcal{V}_D^{(r)}) = |\mathcal{V}_D^{(r)}| \exp\left(-\frac{1}{\mathrm{FDR}^{(r)}}\right) .$$

Instead of $\mathrm{FDR}^{(r)}$, the exponent is $-1/\mathrm{FDR}^{(r)}$ to avoid pathological cases where $\mathrm{FDR}^{(r)}$ approaches $+\infty$, e.g., when all points in a cluster are very concentrated so that $(\bar{\sigma}_k^{(r)})^2 \approx 0$.

Alg. 2.1 incurs overall complexity $\mathcal{O}(NKR_{\max}\check{d}I)$, where I is an upper bound on the number of iterations needed for K-means to converge in step 3, plus $\mathcal{O}(NKR_{\max}\check{d}')$ required in step 6 of Alg. 2.1. Parameters $\check{d}$, $\check{d}'$, and $R_{\max}$ are selected depending on the available computational resources; $(\check{d}, \check{d}')$ should be such that running the computations of Alg. 2.1 on $(\check{d} + \check{d}')$-dimensional vectors can be affordable by the processing unit used. A probabilistic argument for choosing $R_{\max}$ can be determined as elaborated next.

Remark 2. *Using parameters that can be obtained in practice, it is possible to relate the number of random draws $R_{\max}$ with the reliability of SkeVa-based clustering, along the lines of analyzing RANSAC [25].*

To this end, let p denote the probability of having out of R SkeVa realizations at least one "good draw" of $\check{d}$ "informative" dimensions, meaning one for which K-means yields data-cluster associations close to those found by K-means on the full set of D dimensions. Parameter p is a function of the underlying cluster characteristics. It can be selected by the *user and reflects one's level of SkeVa-based big data clustering reliability, e.g.,* $p := 0.95$*. The probability of having all "bad draws" after* R *SkeVa repetitions is clearly* $1-p$*. Moreover, let* q *denote the probability that a randomly drawn row of* $\boldsymbol{X}$ *is "informative." In other words,* q *quantifies prior information on the number of rows (out of* D*) that carry high discriminative information for* K*-means. For instance,* q *can be practically defined by the leverage scores of* $\boldsymbol{X}$*, which typically rank the importance of rows of* $\boldsymbol{X}$ *in large-scale data analytics [18, 20]. An estimate of the leverage scores across the rows of* $\boldsymbol{X}$ *expresses* q *as the percentage of informative rows. Alternatively, if* $\boldsymbol{x}_{n_j} = \boldsymbol{m}_k + \boldsymbol{\Sigma}_k^{1/2}\boldsymbol{v}_{n_j}$ *is the data generation mechanism per cluster* k*, with* $\boldsymbol{v}_{n_j} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_D)$*, then the* i*th entry of* $\boldsymbol{x}_{n_j}$ *is* $\boldsymbol{e}_i^\top \boldsymbol{x}_{n_j} \sim \mathcal{N}(\boldsymbol{e}_i^\top \boldsymbol{m}_k, [\boldsymbol{\Sigma}_k]_{ii})$*. Thus, rows of* $\boldsymbol{X}$ *are realizations of a Gaussian* $1 \times N$ *random vector. If these rows are clustered in* K' *groups,* q *can capture the probability of having an "informative" row located within a confidence region around its associated centroid that contains a high percentage* $\alpha \in (0,1)$ *of its pdf mass. Per SkeVa realization, the probability of drawing* $\check{d}$ *"non-informative" rows can be approximated by* $(1-q)^{\check{d}}$*. Due to the independence of SkeVa realizations,* $1-p = (1-q)^{\check{d}R}$*, which implies that* $R \simeq \log(1-p)/[\check{d}\log(1-q)]$*. Clearly, as* R *increases there is (a growing) nonzero probability that the correct clusters will be revealed. On the other hand, it must be acknowledged that if* R *is not sufficient, clustering performance will suffer commensurately.*

It is interesting to note that as in [25], R *only depends implicitly on* D*, since* q *is a percentage, and does not depend on the validation metric or pertinent thresholds and bounds.*

2.2.2 Sequential algorithm

Drawing a batch of $\check{d}'$ features (rows of $\boldsymbol{X}$) during the validation phase of Alg. 2.1 to assess the discriminating ability of the features drawn in the sketching phase may be computationally, especially if $\check{d}'$ is relatively large. The computation of all distances $\{\delta(\bar{\boldsymbol{x}}_n^{(r)}, \bar{\boldsymbol{c}}_k^{(r)})\}$ in step 6 of Alg. 2.1 can be prohibitive if $\check{d}'$ becomes large. This motivates a sequential augmentation of dimensions, where features are added one at a time, and computations are performed on only a single row of $\boldsymbol{X}$ per feature augmentation, till the upper-bound $\check{d}'$ is reached. Apparently, such an approach adds flexibility and effects computational savings

Algorithm 2.2 Sequential (Se)SkeVa K-means.

```
Input: Data $\boldsymbol{X}$; number of clusters $K$; reduced dimensions $\check{d}$ and $\check{d}'$ of reduced dimensions
    for the sketching and validation phases, respectively; ranking function $f$; number of
    realizations (draws) $R_{\max}$.
Output: Data-cluster associations on $\boldsymbol{X}$.
 1: for $r = 1$ to $R_{\max}$ do
 2:   Randomly sample $\check{d} \ll D$ rows of $\boldsymbol{X}$ to obtain $\check{\boldsymbol{X}}^{(r)}$.
 3:   Run $K$-means on $\check{\boldsymbol{X}}^{(r)}$ to obtain clusters $\{\check{\mathcal{C}}_k^{(r)}\}_{k=1}^K$ and centroids $\{\check{\boldsymbol{c}}_k^{(r)}\}_{k=1}^K$.
 4:   Initialize the auxiliary set of dimensions $\mathcal{A} = \emptyset$.
 5:   for $j = 1$ to $\check{d}'$ do
 6:     Randomly sample 1 dimension of $\boldsymbol{X}$ (other than those in step 2 and in $\mathcal{A}$) to obtain
        row $\check{\mathbf{x}}^{(r')}$ of $\boldsymbol{X}$.
 7:     Include sampled dimension in $\mathcal{A}$.
 8:     Form $\{\bar{\boldsymbol{c}}_k^{(r)} := [\check{\boldsymbol{c}}_k^{(r)\top}, \check{c}_k^{(r')}]^\top\}_{k=1}^K$ as in Alg. 2.1.
 9:     Associate $\{\bar{\boldsymbol{x}}_n^{(r)}\}_{n=1}^N$ to closest $\{\bar{\boldsymbol{c}}_k^{(r)}\}_{k=1}^K$.
10:     Identify validation set $\mathcal{V}_D^{(r)}$ [cf. (2.5)].
11:     if $f(\mathcal{V}_D^{(r)}) < f_{\max}^{(r)}$ or $|\nabla f(\mathcal{V}_D^{(r)})| \leq \epsilon$ then
12:       Go to step 2.
13:     end if
14:   end for
15:   $f_{\max}^{(r+1)} = f(\mathcal{V}_D^{(r)})$.
16:   $r_* = r$.
17: end for
18: Data-cluster associations on $\boldsymbol{X}$ according to $\{\check{\mathcal{C}}_k^{(r_*)}\}_{k=1}^K$.
```

since sequential augmentation of dimensions does not need to be carried out till $\check{d}'$ is reached, but it may be terminated early on if a prescribed criterion is met. These considerations prompted the development of Alg. 2.2.

The sketching phase of Alg. 2.2 remains the same as in Alg. 2.1. In the validation phase, and for each dimension in the additional $\check{d}'$ ones, $\{\check{\boldsymbol{c}}_k^{(r')}\}_{k=1}^K$ are obtained as in Alg. 2.1 [cf. (2.4)], and likewise for $\mathcal{V}_D^{(r)}$. If $f(\mathcal{V}_D^{(r)})$ is smaller than the current maximum value $f_{\max}^{(r)}$ in memory, the $\check{d}$-dimensional clustering $\{\check{\mathcal{C}}_k^{(r)}\}$ is discarded, and a new draw is taken. This can be seen as a "bail-out" test, to reject possibly "bad clusterings" in time, without having to perform the augmentation using all $\check{d}'$ dimensions.

Experiments corroborate that it is not necessary to augment all $D - \check{d}$ dimensions (cf. Fig. 2.1), but using a small subset of them provides satisfactory accuracy while reducing complexity. An alternative route is to stop augmentation once the "gradient" of f, meaning finite differences across augmented dimensions, drops below a prescribed $\epsilon > 0$; that is, $|\nabla f(\mathcal{V}_D^{(r)})| \leq \epsilon$. The sequential approach is summarized in Alg. 2.2, and has complexity strictly smaller than $\mathcal{O}[NKR_{\max}(\check{d}I + \check{d}')]$.

Remark 3. *Using N in the place of D whenever $D \ll N$, or equivalently, replacing $\boldsymbol{X} \in \mathbb{R}^{D\times N}$ with $\boldsymbol{X}^\top$, both the batch and sequential schemes developed for $D \gg$ can be implemented verbatim for $N \gg$. This variant of SkeVa K-means will be detailed in the next subsection for nonlinearly separable clusters.*

2.2.3 Big data kernel clustering

A prominent approach to clustering or classifying nonlinearly separable data is through *kernels;* see e.g., [5]. Vectors $\{\boldsymbol{x}_n\}_{n=1}^N$ are mapped to $\{\varphi(\boldsymbol{x}_n)\}_{n=1}^N$ that live in a higher (possibly infinite-) dimensional space $\mathcal{H}$, where inner products defining distances in $\mathcal{H}$, using the induced norm $\|\cdot\|_{\mathcal{H}} := \langle\cdot \mid \cdot\rangle_{\mathcal{H}}^{1/2}$, are given by a pre-selected (reproducing) kernel function κ; that is, $\langle\varphi(\boldsymbol{x}_n) \mid \varphi(\boldsymbol{x}_{n'})\rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$ [5]. An example of such a kernel is the Gaussian one: $\kappa_{\boldsymbol{\Sigma}}(\boldsymbol{x}_n, \boldsymbol{x}) := \exp[-(\boldsymbol{x} - \boldsymbol{x}_n)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{x}_n)/2]/[(2\pi)^{D/2}(\det \boldsymbol{\Sigma})^{1/2}]$.

For simplicity in exposition, the novel kernel-based (Ke)SkeVa K-means approach to big data clustering will be developed for the hard K-means. Extensions to kernel-based

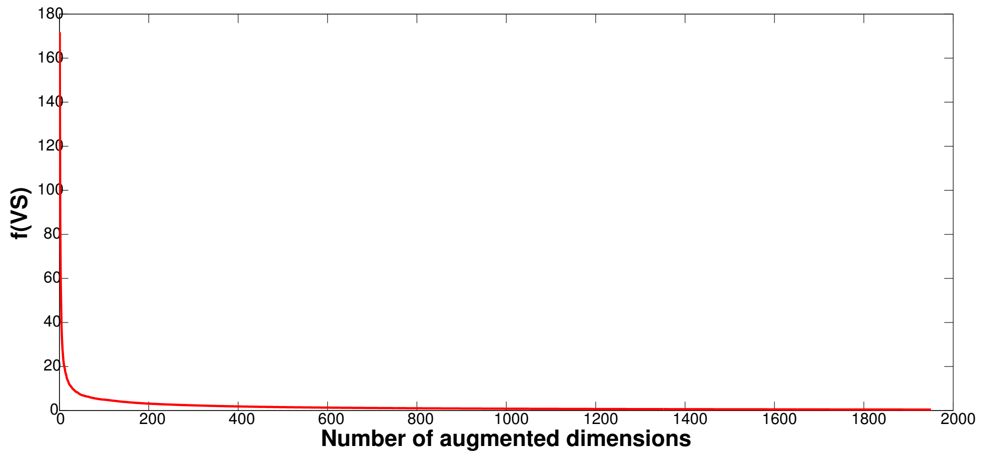


Figure 2.1: $f(\mathcal{V}_D^{(r)})$ vs. the number of augmented dimensions $\check{d}'$ for a synthetic data-set, with $D = 2,000$, $N = 1,000$, $\check{d} = 50$, and full-rank data-model (cf. Secs. 2.2.2 and 2.4).

soft SkeVa K-means follow naturally, and are outlined in Appendix B. Similar to (2.1), the kernel-based hard K-means proceeds as follows. For $i \in \{1, 2, \ldots, I\}$,

$[i$-a$]$ **Update data-cluster associations:** For $n = 1, \ldots, N$,

$$\boldsymbol{x}_n \in \mathcal{C}_k[i] \Leftrightarrow k \in \arg \min_{k' \in \{1,\ldots,K\}} \|\varphi(\boldsymbol{x}_n) - \boldsymbol{c}_{k'}[i]\|_{\mathcal{H}}^2 \tag{2.8a}$$

$[i$-b$]$ **Update cluster centroids:** For $k = 1, \ldots, K$,

$$\boldsymbol{c}_k[i+1] \in \arg\min_{\boldsymbol{c} \in \mathcal{H}} \sum\nolimits_{\boldsymbol{x}_n \in \mathcal{C}_k[i]} \|\varphi(\boldsymbol{x}_n) - \boldsymbol{c}\|_{\mathcal{H}}^2 = \frac{1}{|\mathcal{C}_k[i]|} \sum\nolimits_{\boldsymbol{x}_n \in \mathcal{C}_k[i]} \varphi(\boldsymbol{x}_n) \tag{2.8b}$$

where Euclidean norms $\|\cdot\|_2$ in the standard form of K-means have been replaced by $\|\cdot\|_{\mathcal{H}}$. As the potentially infinite-size $\{\boldsymbol{c}_k[i+1]\}_{k=1}^K$ cannot be stored in memory, step (2.8b) is implicit. In fact, only κ and the data-cluster associations suffice to run (2.8). To illustrate this, substitute $\{\boldsymbol{c}_k[i+1]\}_{k=1}^{K}$ from (2.8b) into (2.8a) to write

$$\begin{aligned}
&\left\| \varphi(\boldsymbol{x}_n) - \frac{1}{\left|\mathcal{C}_{k'}[i+1]\right|} \textstyle\sum_{\boldsymbol{x}'_n \in \mathcal{C}_{k'}[i+1]} \varphi(\boldsymbol{x}'_n) \right\|_{\mathcal{H}}^2 \\
&= \langle \varphi(\boldsymbol{x}_n) \mid \varphi(\boldsymbol{x}_n) \rangle_{\mathcal{H}} \qquad (2.9) \\
&\quad - \frac{2}{\left|\mathcal{C}_{k'}[i+1]\right|} \sum_{\boldsymbol{x}'_n \in \mathcal{C}_{k'}[i+1]} \langle \varphi(\boldsymbol{x}_n) \mid \varphi(\boldsymbol{x}'_n) \rangle_{\mathcal{H}} \\
&\quad + \frac{1}{\left|\mathcal{C}_{k'}[i+1]\right|^2} \sum_{(\boldsymbol{x}'_n, \boldsymbol{x}''_n) \in (\mathcal{C}_{k'}[i+1])^2} \langle \varphi(\boldsymbol{x}'_n) \mid \varphi(\boldsymbol{x}''_n) \rangle_{\mathcal{H}} \\
&= \kappa(\boldsymbol{x}_n, \boldsymbol{x}_n) - \frac{2}{\left|\mathcal{C}_{k'}[i+1]\right|} \sum_{\boldsymbol{x}'_n \in \mathcal{C}_{k'}[i+1]} \kappa(\boldsymbol{x}_n, \boldsymbol{x}'_n) \\
&\quad + \frac{1}{\left|\mathcal{C}_{k'}[i+1]\right|^2} \sum_{(\boldsymbol{x}'_n, \boldsymbol{x}''_n) \in (\mathcal{C}_{k'}[i+1])^2} \kappa(\boldsymbol{x}'_n, \boldsymbol{x}''_n) \, . \qquad (2.10)
\end{aligned}$$

Having established that distances involved in SkeVa K-means are expressible in terms of the chosen kernel κ, the resulting iterative scheme is listed as Alg. 2.3. After randomly selecting an affordable subset $\check{\boldsymbol{X}}^{(r)}$, comprising $\check{\nu}$ columns of $\boldsymbol{X}$ per realization r, and similar to the trial-and-error step in line 3 of Alg. 2.1, the (kernel) K-means of (2.8) is applied to $\check{\boldsymbol{X}}^{(r)}$. The validation phase of KeSkeVa K-means is initialized in line 4, where a second subset $\check{\boldsymbol{X}}^{(r')}$ comprising $\check{\nu}'$ columns from $\boldsymbol{X} \setminus \check{\boldsymbol{X}}^{(r)}$. The distances between data $\{\varphi(\boldsymbol{x}_n^{(r')})\}$ and centroids $\{\check{\boldsymbol{c}}_k^{(r)}\}$ involved in step 5 of Alg. 2.3 are also obtained through kernel evaluations [cf. (2.10)]. This KeSkeVa that operates on the number of data-points rather than dimensions follows along the line of RANSAC [25] but with two major differences: (i) Instead of robust parameter regression, it is tailored for big data clustering; and (ii) rather than consensus it deals with affordably small validation sets across possibly huge data-sets.

During the validation phase, clusters $\check{\mathcal{C}}_k^{(r')}$ are specified according to $\boldsymbol{x}_n^{(r')} \in \check{\mathcal{C}}_k^{(r')} \Leftrightarrow k \in \arg\min_{k' \in \{1,\ldots,K\}} \|\varphi(\boldsymbol{x}_n^{(r')}) - \check{\boldsymbol{c}}_{k'}^{(r)}\|_{\mathcal{H}}^2$. Gathering all information from draws (r, r'), the augmented clusters $\bar{\mathcal{C}}_k^{(r)} := \check{\mathcal{C}}_k^{(r)} \cup \check{\mathcal{C}}_k^{(r')}$ (step 5 of Alg. 2.3) lead to centroids

$$\bar{\boldsymbol{c}}_k^{(r)} := \frac{1}{|\bar{\mathcal{C}}_k^{(r)}|} \sum_{\boldsymbol{x}_n \in \bar{\mathcal{C}}_k^{(r)}} \varphi(\boldsymbol{x}_n) \, . \qquad (2.11)$$

Given the “implicit centroids” obtained as in (2.11), data $\boldsymbol{X}^{(r)}$ are mapped to clusters $\{\bar{\mathcal{C}}_k^{(r)}\}$ which are different from $\check{\mathcal{C}}_k^{(r)}$. To assess this difference, the distance between $\varphi(\boldsymbol{x}_n^{(r)}$

and $\bar{\boldsymbol{c}}_k^{(r)}$ is computed, and columns of $\boldsymbol{X}^{(r)}$ are re-grouped in clusters $\{\check{\bar{\mathcal{C}}}_k^{(r)}\}_{k=1}^K$ as

$$\boldsymbol{x}_n^{(r)} \in \check{\bar{\mathcal{C}}}_k^{(r)} \Leftrightarrow k \in \arg \min_{k' \in \{1,\ldots,K\}} \left\| \varphi\big(\boldsymbol{x}_n^{(r)}\big) - \bar{\boldsymbol{c}}_{k'}^{(r)} \right\|_{\mathcal{H}}^2 . \tag{2.12}$$

Recall that distances are again obtained through kernel evaluations [cf. (2.10)].

The process of generating clusters and centroids in KeSkeVa K-means can be summarized as follows: (i) Group randomly drawn data $\boldsymbol{X}^{(r)}$ into clusters $\check{\mathcal{C}}_k^{(r)}$ with centroids $\check{\boldsymbol{c}}_k^{(r)}$; (ii) draw additional data-points, augment clusters $\bar{\mathcal{C}}_k^{(r)}$, and compute new centroids $\bar{\boldsymbol{c}}_k^{(r)}$; (iii) given $\bar{\boldsymbol{c}}_k^{(r)}$, find clusters $\check{\bar{\mathcal{C}}}_k^{(r)}$ as in (2.12). Since $\check{\boldsymbol{c}}_k^{(r)} \neq \bar{\boldsymbol{c}}_k^{(r)}$ in general, data belonging to $\check{\mathcal{C}}_k^{(r)}$ do not necessarily belong to $\check{\bar{\mathcal{C}}}_k^{(r)}$, and vice versa; while data common to $\check{\bar{\mathcal{C}}}_k^{(r)}$ and $\check{\mathcal{C}}_k^{(r)}$, that is data that have not changed "cluster membership" during the validation phase, comprise the *validation set*

$$\mathcal{V}_N^{(r)} := \left\{ \boldsymbol{x}_n^{(r)} \in \check{\boldsymbol{X}}^{(r)} \;\middle|\; \exists k \text{ s.t. } \boldsymbol{x}_n^{(r)} \in \left(\check{\mathcal{C}}_k^{(r)} \cap \check{\bar{\mathcal{C}}}_k^{(r)} \right) \right\} . \tag{2.13}$$

Among $R_{\max}$ realizations, trial r_* with the highest cardinality $|\mathcal{V}_N^{(r_*)}|$ is identified in Alg. 2.3, and data are finally associated with clusters $\{\check{\mathcal{C}}_k^{(r_*)}\}_{k=1}^K$. The overall complexity of Alg. 2.3 is $\mathcal{O}(DKR_{\max}\check{\nu}^2 I)$ in step 3, when $\{\kappa(\boldsymbol{x}_n, \boldsymbol{x}_{n'})\}_{n,n'=1}^N$ are not stored in memory and kernel evaluations have to be performed for all employed data per realization, plus $\mathcal{O}(DR_{\max}\check{\nu}\check{\nu}')$ in step 5. If $\{\kappa(\boldsymbol{x}_n, \boldsymbol{x}_{n'})\}_{n,n'=1}^N$ are stored in memory, then Alg. 2.3 incurs complexity $\mathcal{O}(KR_{\max}\check{\nu}^2 I + R_{\max}\check{\nu}\check{\nu}')$, which is quadratic only in the small cardinality $\check{\nu}$.

One remark is now in order.

Remark 4. *Similar to all kernel-based approaches, a critical issue is selecting the proper kernel – more a matter of art and prior information about the data. Nonetheless, practical so-termed* multi-kernel *approaches adopt a dictionary of kernels from which one or a few are selected to run K-means on [32]. It will be interesting to investigate whether such multi-kernel approaches can be adapted to the SkeVa-based operation to alleviate the dependence on a single kernel function.*

Algorithm 2.3 Kernel (Ke)SkeVa K-means

```
Input: Data $\boldsymbol{X}$; number of clusters $K$; number $\check{\nu}$ and $\check{\nu}'$ of data during sketching and
    validation phase, respectively; number of realizations $R_{\max}$.
Output: Data-cluster associations on $\boldsymbol{X}$.
 1: for $r = 1$ to $R_{\max}$ do
 2:     Randomly sample $\check{\nu} \ll N$ columns of $\boldsymbol{X}$ to obtain $\check{\boldsymbol{X}}^{(r)}$.
 3:     Apply $K$-means [cf. (B.2) and (2.2)] on $\{\varphi(\boldsymbol{x}_n^{(r)})\}_{n=1}^{\check{\nu}}$; obtain clusters $\{\check{\mathcal{C}}_k^{(r)}\}_{k=1}^{K}$ and
        centroids $\{\check{\boldsymbol{c}}_k^{(r)}\}_{k=1}^{K}$.
 4:     Randomly select $\check{\nu}' \ll N$ columns of $\boldsymbol{X}$, other than those of step 2, to obtain $\check{\boldsymbol{X}}^{(r')}$.
 5:     Associate $\{\varphi(\boldsymbol{x}_n^{(r')})\}_{n=1}^{\check{\nu}'}$ to the closest centroids $\{\check{\boldsymbol{c}}_k^{(r)}\}_{k=1}^{K}$; obtain clusters $\{\bar{\mathcal{C}}_k^{(r)}\}_{k=1}^{K}$
        and centroids $\{\bar{\boldsymbol{c}}_k^{(r)}\}_{k=1}^{K}$ [cf. (2.11)].
 6:     Obtain clusters $\{\check{\bar{\mathcal{C}}}_k^{(r)}\}$ on $\check{\boldsymbol{X}}^{(r)}$ according to (2.12).
 7:     Identify the validation set $\mathcal{V}_N^{(r)}$.
 8: end for
 9: $r_* = \arg\min_{r\in\{1,\ldots,R_{\max}\}} |\mathcal{V}_N^{(r)}|$.
10: Associate $\{\varphi(\boldsymbol{x}_n)\}_{n=1}^{N}$ according to the closest centroids obtained from clusters
    $\{\check{\mathcal{C}}_k^{(r_*)}\}_{k=1}^{K}$.
```

2.3 Divergence-Based SkeVa

A limitation of Algs. 2.1 and 2.2 is the trial-and-error strategy which requires clustering per random draw of samples. This section introduces a method to surmount such a need and select a small number of data or dimensions on which only a *single* clustering step is applied at the last stage of the algorithm. As the "quality" per draw is assessed without clustering, this approach trades off accuracy for reduced complexity.

2.3.1 Large-scale data sets

First, the case where random draws are performed on the N data will be examined, followed by draws across the D dimensions. Any randomly drawn data from $\{\boldsymbol{x}_n\}_{n=1}^N$ will be assumed centered around their sample mean.

Since intermediate clustering will not be applied to assess the quality of a random draw, a metric is needed to quantify how well the randomly drawn samples represent clusters. To this end, motivated by the pdf mixture model [cf. (2.3)], consider the following pdf estimate formed using the randomly selected data

$$\check{p}^{(r)}(\boldsymbol{x}) := \frac{1}{\check{\nu}} \sum_{n=1}^{\check{\nu}} \kappa\big(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}\big) \tag{2.14}$$

where κ stands for a user-defined kernel function, parameterized by $\boldsymbol{x}_n^{(r)}$, and satisfying $\int \kappa(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}) d\boldsymbol{x} = 1$ for the estimate in (2.14) to qualify as a pdf. Here, the Gaussian kernel $\kappa_{\boldsymbol{\Sigma}}(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}) := \exp[-(\boldsymbol{x} - \boldsymbol{x}_n^{(r)})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{x}_n^{(r)})/2]/[(2\pi)^{D/2}(\det \boldsymbol{\Sigma})^{1/2}]$ is considered with $\boldsymbol{\Sigma} := \sigma^2 \boldsymbol{I}_D$.

Having linked random samples with pdf estimates, the assessment whether a draw represents well the whole population $\{\boldsymbol{x}_n\}_{n=1}^N$ translates to how successful this draw is in estimating the actual data pdf via (2.14). A random sample where all selected data form a single cluster is clearly not a good representative of $\{\boldsymbol{x}_n\}_{n=1}^N$. For example, if the selected points are gathered around $\boldsymbol{0}$ (recall that drawn data are centered around their sample mean), then the resulting pdf estimate (2.14) will resemble the uni-modal (thick) dashed curve in Fig. 2.2. Such a pdf estimate is a poor representative of the whole data-set, and

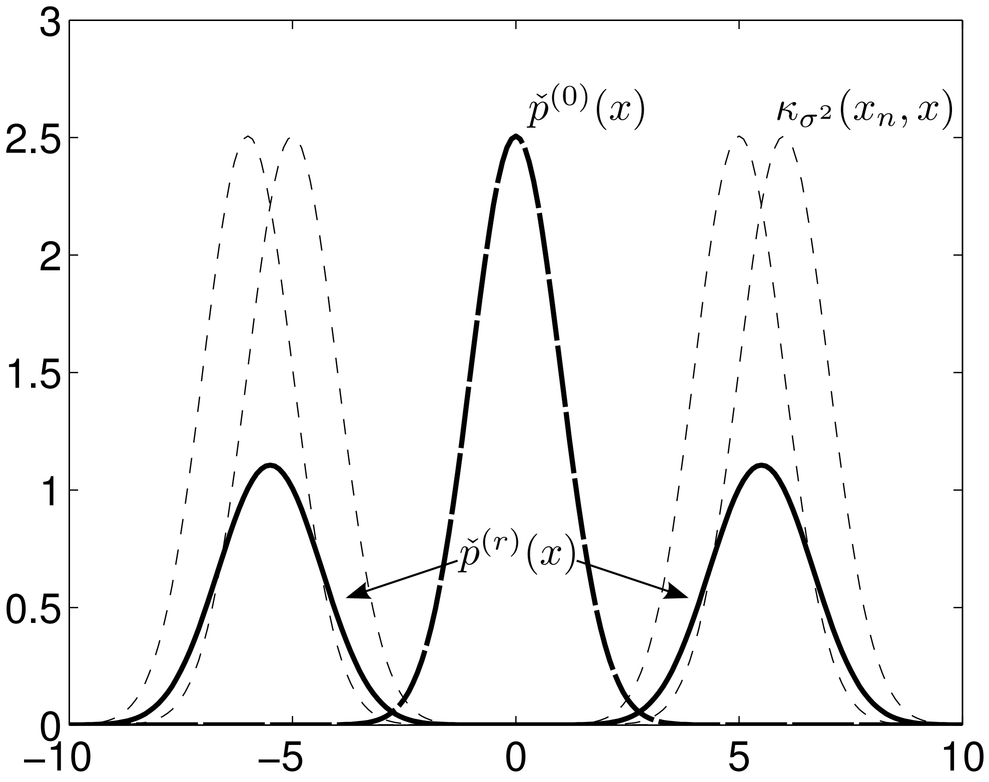


Figure 2.2: Examples of pdf mixtures fitted to data-points. Reference pdf estimate is the uni-modal thick dashed curve; the larger the divergence from this estimate, the larger the probability of producing meaningful clustering.

clustering on that small population of data should be avoided. On the contrary, random draws yielding the multi-modal (thick) solid curve in Fig. 2.2 should be highly rated as potential candidates for clustering. As a *first step* toward assessing draws is a metric quantifying how "far" the pdf $\check{p}^{(r)}$ is from $\check{p}^{(0)}(\boldsymbol{x}) := \kappa(\mathbf{0}, \boldsymbol{x})$.

Among candidate metrics of "distance" between pdfs, the Cauchy-Schwarz divergence [33] is chosen here:

$$\Delta_{\mathrm{CS}}\big(\check{p}^{(r)} \,\|\, \check{p}^{(r')}\big) := -\log \frac{\left(\int \check{p}^{(r)}(\boldsymbol{x})\check{p}^{(r')}(\boldsymbol{x})d\boldsymbol{x}\right)^2}{\int [\check{p}^{(r)}(\boldsymbol{x})]^2 d\boldsymbol{x} \int [\check{p}^{(r')}(\boldsymbol{x})]^2 d\boldsymbol{x}} \,.$$

The reason for choosing Δ_{CS} over other popular divergences, such as the Kullback-Leibler one, is the ease of obtaining pdf estimates via (2.14). Specifically, the numerator in Δ_{CS} can be expressed as [cf. (2.14)]

$$\int \check{p}^{(r)}(\boldsymbol{x})\check{p}^{(r')}(\boldsymbol{x})d\boldsymbol{x} = \frac{1}{\check{\nu}\check{\nu}'}\sum_{n=1}^{\check{\nu}}\sum_{n'=1}^{\check{\nu}'}\int \kappa\big(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}\big)\kappa\big(\boldsymbol{x}_{n'}^{(r')}, \boldsymbol{x}\big)d\boldsymbol{x}\,. \tag{2.15}$$

The right-hand-side of (2.15) is simplified further if the chosen kernel is the Gaussian one. As the convolution of Gaussian pdfs is also Gaussian, it is not hard to verify that

$$\int \kappa_{\boldsymbol{\Sigma}}\big(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}\big)\kappa_{\boldsymbol{\Sigma}}\big(\boldsymbol{x}_{n'}^{(r')}, \boldsymbol{x}\big)d\boldsymbol{x} = \kappa_{2\boldsymbol{\Sigma}}\big(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}_{n'}^{(r')}\big)$$

for which (2.15) becomes

$$\int \check{p}^{(r)}(\boldsymbol{x})\check{p}^{(r')}(\boldsymbol{x})d\boldsymbol{x} = \frac{1}{\check{\nu}\check{\nu}'}\mathbf{1}^\top \boldsymbol{K}_{2\boldsymbol{\Sigma}}^{(r,r')}\mathbf{1}$$

where the $\check{\nu} \times \check{\nu}'$ matrix $\boldsymbol{K}_{2\boldsymbol{\Sigma}}^{(r,r')}$ has (n, n')th entry $[\boldsymbol{K}_{2\boldsymbol{\Sigma}}^{(r,r')}]_{nn'} := \kappa_{2\boldsymbol{\Sigma}}(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}_{n'}^{(r')})$. It thus follows that

$$\begin{aligned}\Delta_{\mathrm{CS}}\big(\check{p}^{(r)} \,\|\, \check{p}^{(r')}\big) = &-2\log\left(\frac{1}{\check{\nu}\check{\nu}'}\mathbf{1}^\top \boldsymbol{K}_{2\boldsymbol{\Sigma}}^{(r,r')}\mathbf{1}\right)\\ &+\log\left(\frac{1}{\check{\nu}^2}\mathbf{1}^\top \boldsymbol{K}_{2\boldsymbol{\Sigma}}^{(r,r)}\mathbf{1}\right)\\ &+\log\left(\frac{1}{\check{\nu}'^2}\mathbf{1}^\top \boldsymbol{K}_{2\boldsymbol{\Sigma}}^{(r',r')}\mathbf{1}\right)\,.\end{aligned} \tag{2.16}$$

Notice that when r' is simply $\{\mathbf{0}\}$, the last summand in (2.16) becomes $\log \kappa_{2\boldsymbol{\Sigma}}(\mathbf{0}, \mathbf{0}) = -D(\log 2\pi)/2 - (\log\det 2\boldsymbol{\Sigma})/2$.

The metric in (2.16) is computed per draw of the so-termed divergence-based DiSkeVa K-means summarized in Alg. 2.4. A number $R_{\max}$ of realizations are attempted to discover a "good" draw r_* of data, to which clustering is finally performed. Line 3 in Alg. 2.4 checks whether the randomly selected subset yield via (2.14) a pdf $\check{p}^{(r)}$ that differs enough from the "singular" pdf $\check{p}^{(0)} = \kappa_{\boldsymbol{\Sigma}}(\mathbf{0}, \cdot)$. If the divergence exceeds $\Delta_{\max}$, realization r will be further explored, otherwise $r+1$ is drawn. Notice that threshold $\Delta_{\max}$ is adaptively defined and takes, according to line 8, the maximum recorded value from realization $r = 0$ till the current one.

If $\check{p}^{(r)}$ passes the first check of being far from $\check{p}^{(0)}$, lines 4 to 10 implement the *second step* of consenting whether r is indeed a "good" realization. To this end, a number of

Algorithm 2.4 Divergence (Di)SkeVa K-means on N.

Input: Data $\boldsymbol{X}$; number of clusters K; number $\check{\nu}$ and $\check{\nu}'$ of points for sketching and validation phases, respectively; number of realizations $R_{\max}$; $\Delta_{\max} = 0$, $\Delta'_{\min} = +\infty$.

Output: Data-cluster associations.

```
1:  for $r = 1$ to $R_{\max}$ do
2:    Let $\check{\boldsymbol{X}}^{(r)}$ denote $\check{\nu}$ randomly selected points after centered around their sample mean.
3:    if $\Delta_{\mathrm{CS}}(\check{p}^{(r)} \,\|\, \check{p}^{(0)}) > \Delta_{\max}$ then
4:      Let $\check{\boldsymbol{X}}^{(r')}$ denote $\check{\nu}'$ randomly selected points, other than those in step 2, after centered around their sample mean.
5:      $\bar{\boldsymbol{X}}^{(r)} := [\check{\boldsymbol{X}}^{(r)}, \check{\boldsymbol{X}}^{(r')}]$.
6:      if $\Delta_{\mathrm{CS}}(\bar{p}^{(r)} \,\|\, \check{p}^{(r')}) < \Delta'_{\min}$ then
7:        $\Delta'_{\min} := \Delta_{\mathrm{CS}}(\bar{p}^{(r)} \,\|\, \check{p}^{(r')})$.
8:        $\Delta_{\max} := \Delta_{\mathrm{CS}}(\check{p}^{(r)} \,\|\, \check{p}^{(0)})$.
9:        $r_* := r$.
10:     end if
11:   end if
12: end for
13: Perform $K$-means on $\check{\boldsymbol{X}}^{(r_*)}$ and associate $\boldsymbol{X}$ to clusters obtained by $K$-means on $\check{\boldsymbol{X}}^{(r_*)}$.
```

$\check{\nu}'$ additional data-points is drawn to form $\bar{\boldsymbol{X}}^{(r)} := [\check{\boldsymbol{X}}^{(r)}, \check{\boldsymbol{X}}^{(r')}]$ in line 5. The mixture pdf $\bar{p}^{(r)}$ corresponding to $\bar{\boldsymbol{X}}^{(r)}$, should stay as close as possible to $\check{p}^{(r)}$ since reliable pdf estimates should remain approximately invariant as extra data are added. Drastic changes of Δ_{CS} before and after augmentation suggests that the draw is likely not to be a good representative of the whole population. Notice here that $\Delta'_{\min}$ is also adaptively defined to take the minimum value among all recorded divergences from the start of iterations. Moreover, both updates of $\Delta'_{\min}$ and $\Delta_{\max}$ are performed once the candidate draw r has passed through the "check-points" of lines 3 and 6.

In the case where the Gaussian kernel is employed, Alg. 2.4 has overall complexity $o[DR_{\max}\check{\nu}^2 + DR_{\max}(\check{\nu}\check{\nu}' + \check{\nu}'^2)]$, if the kernel matrix $\boldsymbol{K}_{2\boldsymbol{\Sigma}}$ of all data $\{\boldsymbol{x}_n\}_{n=1}^N$ is not stored in memory and calculations of all kernel sub-matrices in (2.16) are performed per realization; plus, $\mathcal{O}(D\check{\nu}KI)$ for a single application of K-means on the finally selected draw r_*. If $\boldsymbol{K}_{2\boldsymbol{\Sigma}}$ is available in memory, then Alg. 2.4 incurs complexity $o[R_{\max}\check{\nu}^2 + R_{\max}(\check{\nu}\check{\nu}' + \check{\nu}'^2) + D\check{\nu}KI]$, that is quadratic only in the small subset sizes.

Remark 5. *Along the lines of Remark 2, let p denote the probability of having out of R SkeVa realizations at least one "good draw" of size $\check{\nu}$, meaning one for which K-means yields centroids close to those found with the "full data-set." Moreover, let q denote the probability of a datum to lie "close" to its associated centroid. For example, q can capture the probability of having a datum located within a confidence region which is centered at its associated centroid and contains a high percentage of its pdf mass. The probability of having all "bad draws" is clearly $1-p$. Assuming that data are drawn independently, the probability of having one draw contain only data located "far away" from centroids is $(1-q)^{\check{\nu}}$. Due to the independence of random draws in SkeVa, $1-p = (1-q)^{\check{\nu}R}$, which implies that $R \simeq \log(1-p)/[\check{\nu}\log(1-q)]$. Analogous to Remark 2, this argument neither involves N nor it depends on the validation metric or pertinent thresholds and bounds.*

2.3.2 High-dimensional data

Alg. 2.4 remains operational also when DiSkeVa K-means deals with $D \gg$. Although the proposed scheme can be generalized to cope with both $N \gg$ and $D \gg$, for simplicity of exposition it will be assumed that only $D \gg$. To this end, consider the following pdf estimate $\mathbb{R}^{\check{d}}$:

$$\check{p}^{(r)}(\check{\boldsymbol{x}}) := \frac{1}{N}\sum_{n=1}^{N} \kappa\big(\check{\boldsymbol{x}}_n^{(r)}, \check{\boldsymbol{x}}\big)\,, \quad \forall \check{\boldsymbol{x}} \in \mathbb{R}^{\check{d}}$$

where $\check{\boldsymbol{x}}_n^{(r)}$ denotes a $\check{d} \times 1$ subvector of the $D \times 1$ vector $\boldsymbol{x}_n$.

The counterpart of Alg. 2.4 on dimensions is listed as Alg. 2.5. Although along the lines of Alg. 2.4, there is a notable difference. In the validation step, where dimensions are increased (cf. line 6) and a pdf estimate is needed for the augmented set of variables. To define divergence between pdfs of different dimensions, vectors have to be zero padded from the $\check{d}$-dimensional $\check{\boldsymbol{x}}_n^{(r)}$ to the $(\check{d} + \check{d}')$-dimensional $\bar{\boldsymbol{\chi}}_n^{(r)} := [\check{\boldsymbol{x}}_n^{(r)\top}, \mathbf{0}^\top]^\top$. Recall here that $\bar{\boldsymbol{x}}_n^{(r)} := [\check{\boldsymbol{x}}_n^{(r)\top}, \check{\boldsymbol{x}}_n^{(r')\top}]^\top$. To avoid confusion, pdf mixtures on these zero-padded vectors are given by

$$\bar{q}^{(r)}(\bar{\boldsymbol{x}}) = \frac{1}{N}\sum_{n=1}^{N} \kappa\big(\bar{\boldsymbol{\chi}}_n^{(r)}, \bar{\boldsymbol{x}}\big)\,, \quad \forall \bar{\boldsymbol{x}} \in \mathbb{R}^{\check{d}+\check{d}'}\,. \tag{2.17}$$

Similar to Alg. 2.4, the overall complexity of Alg. 2.5 is $o[(\check{d}+\check{d}')N^2 R_{\max}]$ for computations in (2.16), plus $\mathcal{O}(\check{d}NKI)$ for a single application of K-means on the finally selected draw r_*.

Remark 6. *If multi-core machines are also available, the validation phases of Algs. 2.1-2.5 can be readily parallelized, using recent advances on efficient parallel computing platforms such as MapReduce [34, 35].*

2.4 Numerical Tests

The proposed algorithms were validated on synthetic and real data-sets. Tests involve either large number of data ($N \gg$) and/or large number of dimensions ($D \gg$). The following methods were also tested: (i) The standard hard K-means [cf. (2.1)], run on the full range of N data-points and D dimensions, which is abbreviated in the figures as "full K-means"; (ii) the state-of-the-art RP-based *feature-extraction* scheme [20, Alg. 2], with a Bernoulli-type RP matrix as in [36]; (iii) the *randomized feature-selection* (RFS) algorithm [20, Alg. 1; $\epsilon = 1/3$], a leverage-scores-based scheme; and (iv) the *"approximate kernel K-means"* algorithm

Algorithm 2.5 Divergence (Di)SkeVa K-means on D.

```
Input: Data $\boldsymbol{X}$; number of clusters $K$; number $\check{d}$ and $\check{d}'$ of dimensions for sketching and
    validation phases, respectively; number of realizations $R_{\max}$; $\Delta_{\max} = 0$, $\Delta'_{\min} = +\infty$.
Output: Data-cluster associations.
 1: for $r = 1$ to $R_{\max}$ do
 2:   Center randomly selected $\check{\boldsymbol{X}}^{(r)}$ around their sample mean to obtain $\check{\boldsymbol{X}}^{(r)}$.
 3:   if $\Delta_{\text{CS}}(\check{p}^{(r)} \,\|\, \check{p}^{(0)}) > \Delta_{\max}$ then
 4:     Center randomly selected $\check{\boldsymbol{X}}^{(r')}$, other than those in step 2, around their sample
        mean to obtain $\check{\boldsymbol{X}}^{(r')}$.
 5:     Form $\bar{\boldsymbol{X}}^{(r)} := [\check{\boldsymbol{X}}^{(r)\top}, \check{\boldsymbol{X}}^{(r')\top}]^\top$.
 6:     Form $\bar{\boldsymbol{\mathcal{X}}}^{(r)} := [\check{\boldsymbol{X}}^{(r)\top}, \mathbf{0}^\top]^\top$ and estimate pdf mixture $\bar{q}^{(r)}$ [cf. (2.17)].
 7:     if $\Delta_{\text{CS}}(\bar{p}^{(r)} \,\|\, \bar{q}^{(r)}) < \Delta'_{\min}$ then
 8:       $\Delta'_{\min} := \Delta_{\text{CS}}(\bar{p}^{(r)} \,\|\, \bar{q}^{(r)})$.
 9:       $\Delta_{\max} := \Delta_{\text{CS}}(\check{p}^{(r)} \,\|\, \check{p}^{(0)})$.
10:       $r_* := r$.
11:     end if
12:   end if
13: end for
14: Perform $K$-means on $\check{\boldsymbol{X}}^{(r_*)}$ and associate $\boldsymbol{X}$ to clusters obtained by $K$-means on $\check{\boldsymbol{X}}^{(r_*)}$.
```

[16, Alg. 2], which solves for an "optimal" data-cluster association matrix given a randomly selected subset of the original data-points. For fairness, the naive kernel K-means algorithm in [16, Alg. 1] is not tested, because a random draw of data and the application of K-means is done only *once* in [16, Alg. 1]; hence, the attractive attribute of multiple independent draws is not leveraged as in SkeVa. To mitigate initialization-dependent performance, each realization of K-means, including also its usage as a module in other competing methods, is run five times with different initialization per run, keeping finally only the data-clusters association that results with the smallest sum of distances of data from the associated centroids.

As figures of merit the relative clustering accuracy and the execution time (in secs) were adopted. Relative clustering accuracy is defined as the percentage of points assigned to the correct clusters (empirical probability of correct clustering), relative to that of (kernel) K-means on the full data-set. Regarding computational time evaluations, tests in Sec. 2.4.1 are run using Matlab [37] on a SunFire X4600 PC with a 32-core AMD Opteron 8356, clocked at 2.3GHz with 128GB RAM memory [38], without the use of parallelization, on a single computational thread. Tests in Secs. 2.4.2, 2.4.3 and 2.4.4 are run on an HP ProLiant BL280c G6 server using 2 eight-core Sandy Bridge E5-2670 processor chips (2.6GHz) and 128GB of RAM memory [38]. In the latter tests, algorithms were allowed to exploit MATLAB's inherent multithread capabilities [39] on the 16 cores of the server. Moreover, all plotted curves are averages over 50 Monte Carlo realizations.

To construct synthetic data, $D \times 1$ vectors $\{\boldsymbol{x}_n\}_{n=1}^{N}$ were generated according to the following model per cluster k:

$$\boldsymbol{x}_{n_j} = \boldsymbol{m}_k + \boldsymbol{\Sigma}_k^{1/2} \boldsymbol{v}_{n_j}\,, \quad j \in \{1, \ldots, N/K\} \tag{2.18}$$

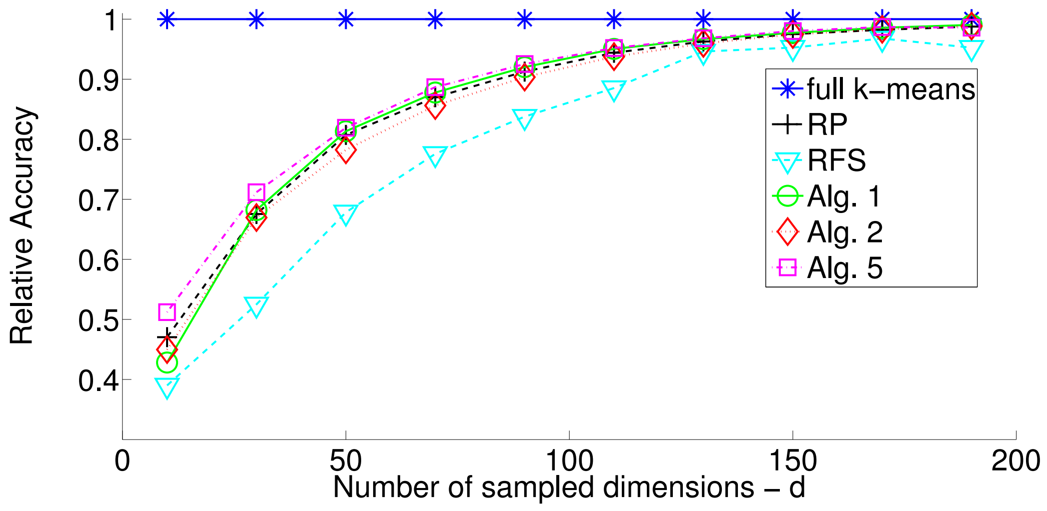
where it is assumed that N is an integer multiple of K, $\boldsymbol{m}_k$ is the $D \times 1$ mean (centroid) of cluster k, noise $\boldsymbol{v}_{n_j} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_D)$ is standardized Gaussian, and $\boldsymbol{\Sigma}_k$ is the covariance matrix of the data generated for cluster k; hence, $\boldsymbol{x}_{n_j} \sim \mathcal{N}(\boldsymbol{m}_k, \boldsymbol{\Sigma}_k)$. Means $\{\boldsymbol{m}_k\}_{k=1}^{K}$ are selected uniformly at random from a D-dimensional hypercube, as in [20]. To accommodate data-models with limited degrees of freedom, the "rank of data," controlled by the number of non-zero eigenvalues of $\boldsymbol{\Sigma}_k$, was used as a tuning parameter. In certain cases, clusters were well separated—a scenario where K-means achieves relatively high clustering accuracy. Throughout this section $R_{\max} = 10$ except for the tests using the KDDb database [40] for which $R_{\max} = 20$.

2.4.1 Large number of dimensions ($D \gg$)

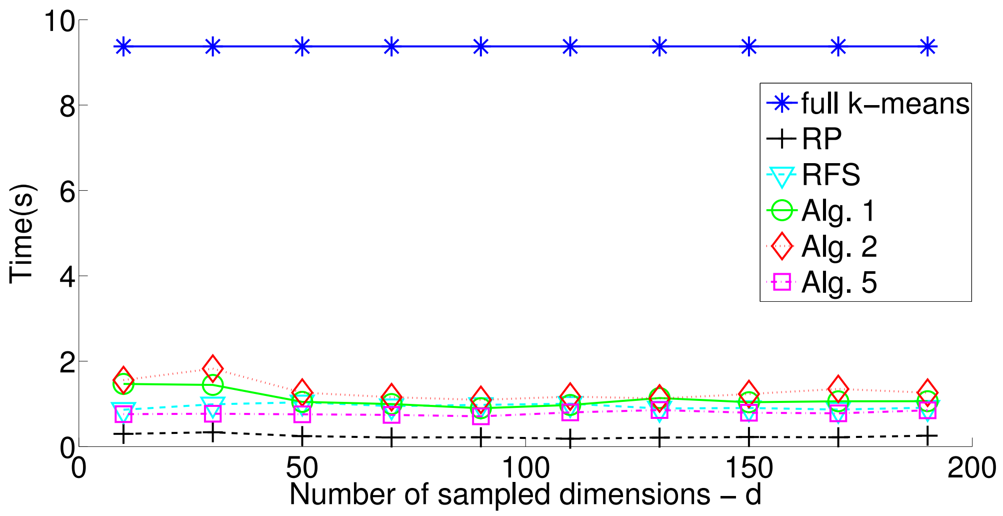
Tests cases in this subsection have $D \gg N$. Competing methods are the "full K-means," RP [20, Alg. 2], and RFS [20, Alg. 1; $\epsilon = 1/3$]. Model (2.18) was used to generate $N = 1,000$ D-dimensional vectors for $K = 5$ clusters, for several values of D, and variable "data-rank." It can be seen from Figs. 2.3 and 2.4 that Algs. 2.1 (SkeVa K-means) and 2.2 (SeSkeVa K-means) approach the accuracy of the full K-means algorithm as the number of sampled dimensions d increases. As expected, computational time is significantly lower than that of full K-means, since the latter operates on all D dimensions. Moreover, SeSkeVa K-means needs more time than RP [20] to achieve the same clustering accuracy (cf. Figs. 2.3 and 2.4), since RP utilizes K-means as a sub-module only once, after dimensionality-reduction has been effected by left-multiplication of $\boldsymbol{X}$ with a (fat) $d \times D$ RP matrix. However, this changes as D grows large. As Fig. 2.5 demonstrates, whenever D is massive, left-multiplying $\boldsymbol{X}$ by the RP $d \times D$ matrix in [20] can become cumbersome, resulting in computational times larger than those of SeSkeVa K-means.

Fig. 2.6 shows results for the real data-set ARCENE [41], which contains mass-spectra of patients diagnosed with cancer, as well as spectra associated with healthy individuals. Clustering involves grouping 100 ($D = 10,000$)-dimensional spectra in two clusters ($K = 2$). The number of augmented dimensions for all employed algorithms is $\check{d}' = 100$. All proposed algorithms approach the performance of RP and full K-means, while requiring less time. Alg. 2.5 is the fastest one at a comparable performance.

Fig. 2.7 depicts results for the real ORL database of 400 face-images, from 40 different subjects (10 each) [42, 43]. Images have size 92×112 with 8-bit grey levels, resulting in $D = 10,304$. Only 30 images (3 subjects) were used, and as such the task is to group these images into $K = 3$ clusters. As with the ARCENE data, the number of additional dimensions for all proposed algorithms is $\check{d}' = 100$. Algs. 2.1, 2.2, and 2.5 exhibit similar

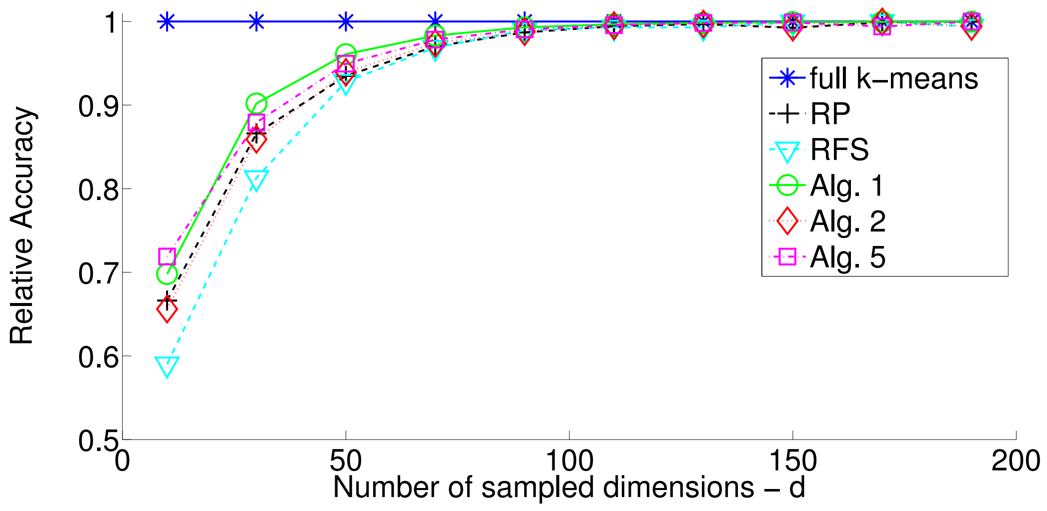


(a) Relative clustering accuracy

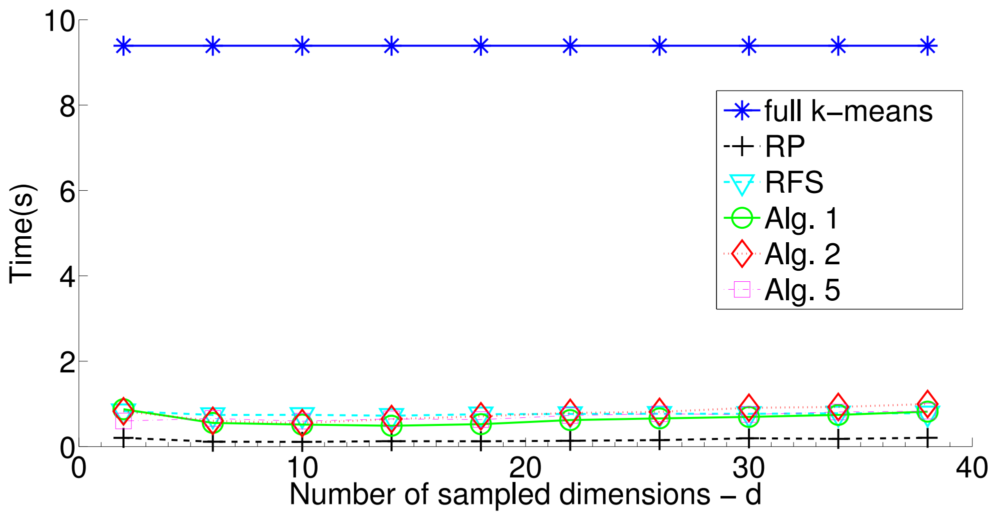


(b) Clustering time (secs)

Figure 2.3: Synthetic data ($D = 2,000$ and full-rank model).

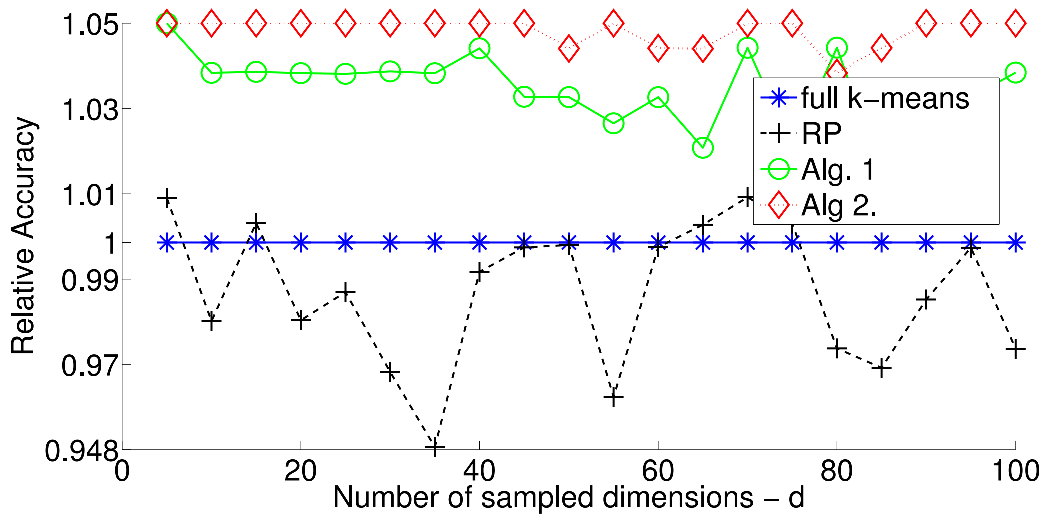


(a) Relative clustering accuracy

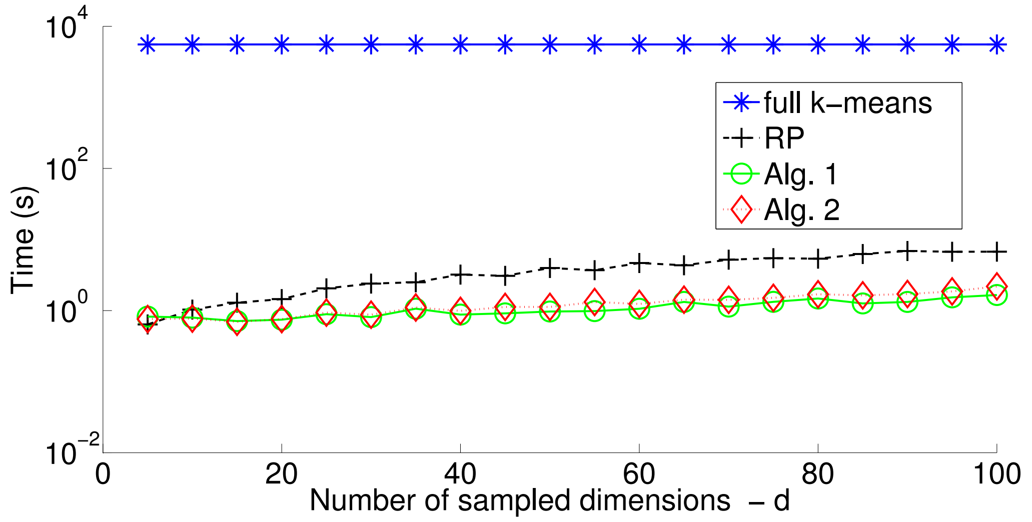


(b) Clustering time (secs)

Figure 2.4: Synthetic data ($D = 2,000$ and rank equal to 500).



(a) Relative clustering accuracy



(b) Clustering time (secs)

Figure 2.5: Synthetic data ($D = 500,000$ and rank equal to $1,000$).

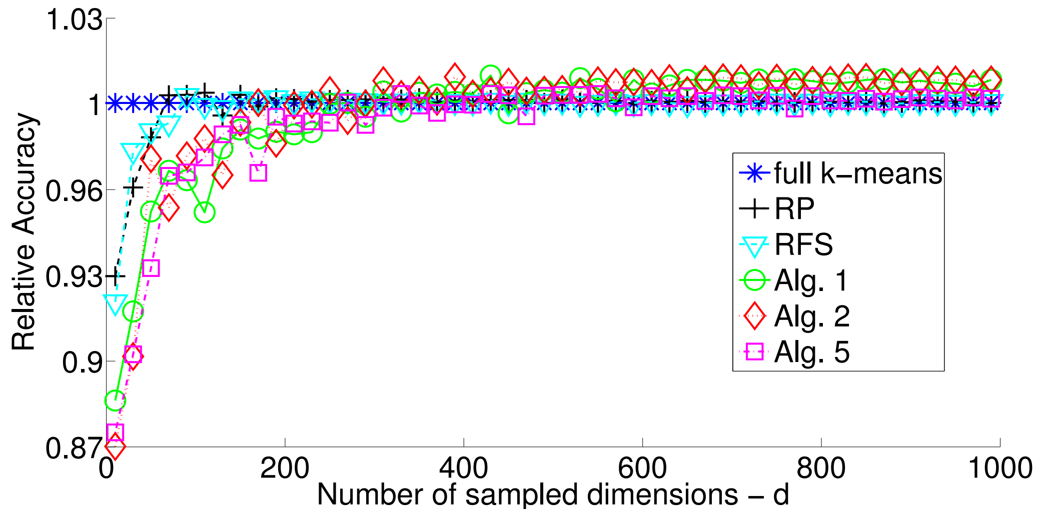
performance, while requiring much less time than the full K-means. Again, Alg. 2.5 is the fastest one at a comparable performance.

Tests were also performed on a subset of the KDDb 2010 data-set ($K = 2$, $D = 2,990,384$) [40]. The version of the data-set is the one transformed by the winner of the KDD 2010 Cup (National Taiwan University). In each run $10,000$ data-points were chosen randomly from both classes, and clustering was performed on this subset. The RFS performance is not reported in Fig. 2.8 as there were issues regarding memory usage due to the required SVD computations. Here, the number of augmented dimensions is $\check{d}' = 1,000$. All algorithms exhibit performance similar to full K-means; however Alg. 2.1 and Alg. 2.5 require significantly less time than all competing alternatives.

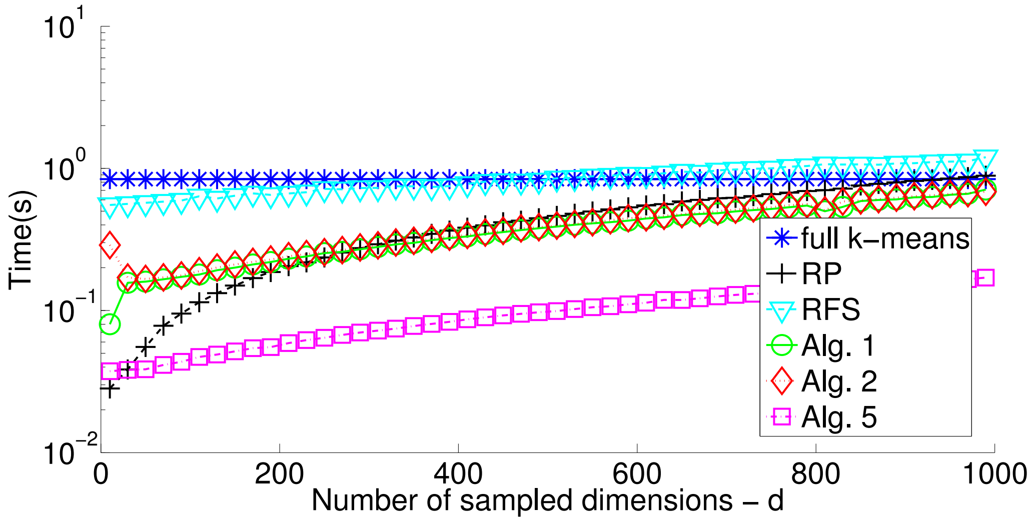
It should be noted also that the required amount of memory per iteration for Algs. 2.1 and 2.2 is at most $\mathcal{O}[N(\check{d} + \check{d}')]$, in contrast with the competing algorithms whose memory requirements grow linearly with D.

2.4.2 Large number of points ($N \gg$)

Here the number of datapoints is much greater than the dimension of the dataset ($N \gg D$). Alg. 2.4 is compared with the "full K-means" and the "approximate kernel K-means" algorithm [16]. $N = 100,000$ vectors with $D = 5$ were generated according to (2.18) for $K = 5$ clusters. Although "approximate kernel K-means" can accommodate nonlinearly separable clusters by using nonlinear kernel functions, the linear kernel was used here: $\kappa(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{x}^\top \boldsymbol{y}$, $\forall \boldsymbol{x}, \boldsymbol{y}$. The Gaussian kernel function $\kappa_{\boldsymbol{\Sigma}}$, with $\boldsymbol{\Sigma} := \boldsymbol{I}_D$, was used in (2.14). Fig. 2.9 shows clustering accuracy across the number $\check{\nu}$ of randomly selected data per draw. The number of additional points $\check{\nu}'$ is set equal to 100. As Fig. 2.9 demonstrates, Alg. 2.4 approaches the performance of the full K-means algorithm, even with $\check{\nu} = 100$ sampled data, while requiring markedly lower execution time than both "full" and "approximate kernel K-means."

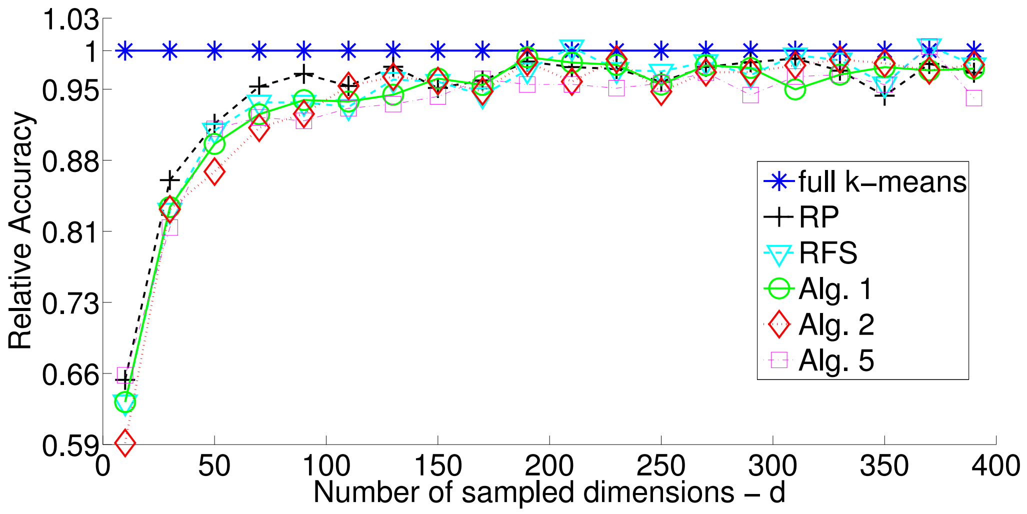


(a) Relative clustering accuracy

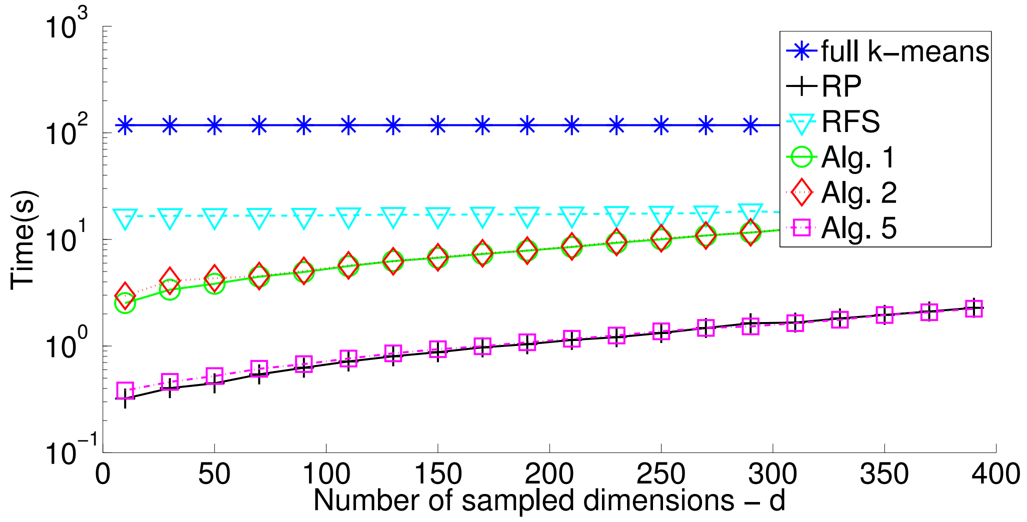


(b) Clustering time (secs)

Figure 2.6: Simulated performance for real data-set ARCENE.

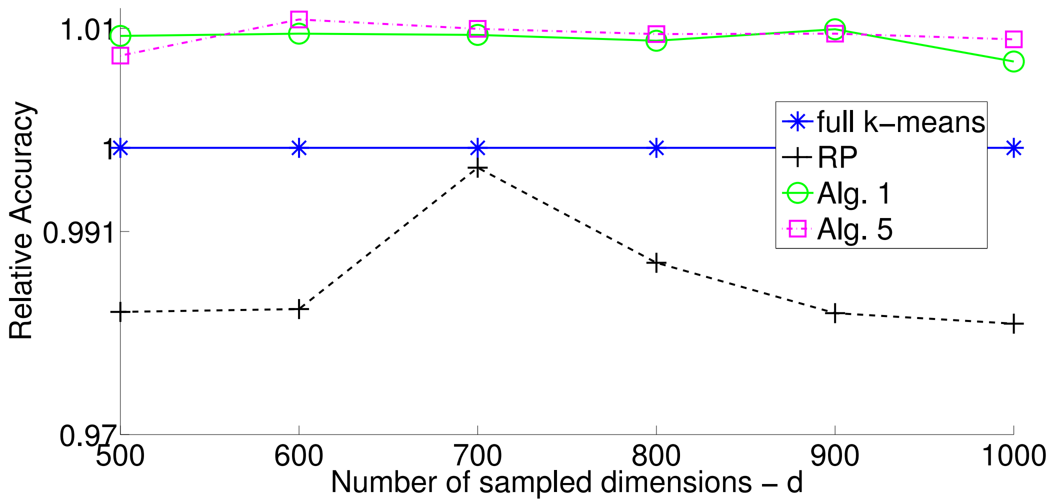


(a) Relative clustering accuracy

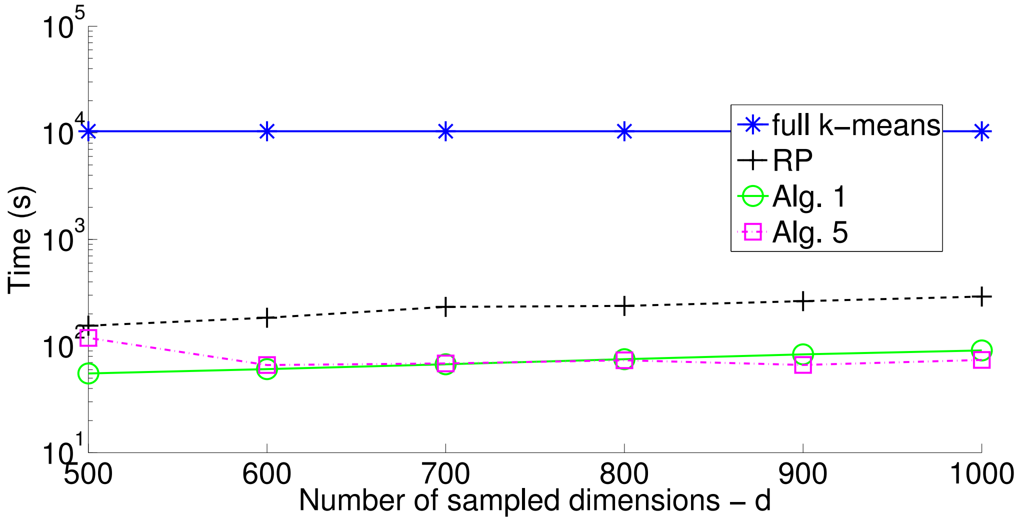


(b) Clustering time (secs)

Figure 2.7: Simulated performance for real data-set ORL.



(a) Relative clustering accuracy



(b) Clustering time (secs)

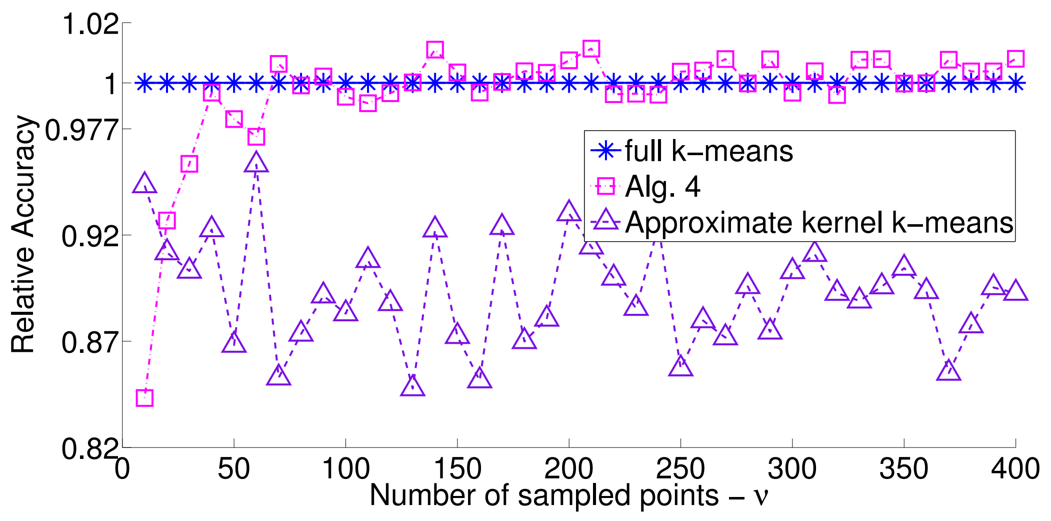
Figure 2.8: Simulated performance for real data-set KDDb.

2.4.3 Kernel clustering

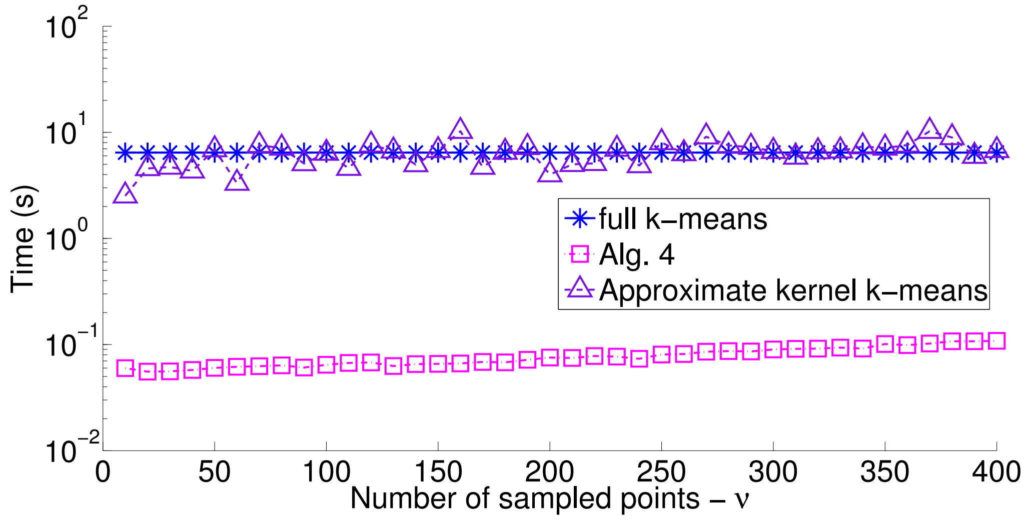
Nonlinearly separable data are mapped here using a prescribed kernel function to high-dimensional spaces to render them linearly separable. Algs. 2.3 and 2.4, with kernel K-means applied only at the end, are compared with the "full kernel K-means" and the "approximate kernel K-means" [16]. Throughout, $\boldsymbol{\Sigma} := 5\boldsymbol{I}_D$ was used in (2.14). Tests were performed on a subset ($N = 35,000$) of the MNIST-784 data-set, which contains 28×28 pixel images of handwritten digits grouped in $K = 10$ clusters. The kernel used in this case is the sigmoid one $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \tanh(\alpha \boldsymbol{x}^\top \boldsymbol{y} + b)$ with parameters $\alpha = 0.0045$, $b = 0.11$, in accordance to [44]. Both the sigmoid and the Gaussian (for the case of Alg. 2.4) kernels are considered stored in memory. Fig. 2.10 depicts the relative clustering accuracy for this data-set and the required time in seconds. It is clear that accuracies of all three algorithms are close and approach the performance of "full kernel K-means" as the number of sampled data-points increases. However, the time required by Algs. 2.3 and 2.4 is significantly less than the time required by the "full" and "approximate kernel K-means."

2.4.4 Exploiting multiple computational threads

To showcase the scalability of the proposed algorithms in the presence of multiple computational nodes, the algorithms were run on multiple computational threads. The independent draws r of the proposed algorithms were executed in parallel. Moreover, competing algorithms were allowed to exploit MATLAB's multithread capabilities, e.g., matrix-matrix multiplications in RP [39]. Figs. 2.11 and 2.12 show simulation results for the ARCENE and KDDb data-sets, respectively. Clearly, parallelization of the iterations on the proposed algorithms is beneficial since the algorithms exhibit about an order of magnitude less required time than that of competing methods.

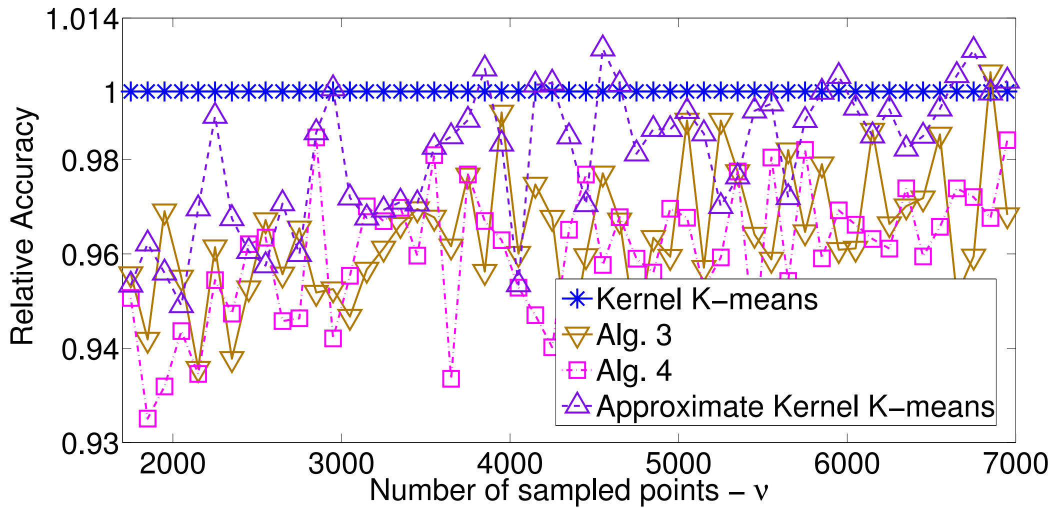


(a) Relative clustering accuracy

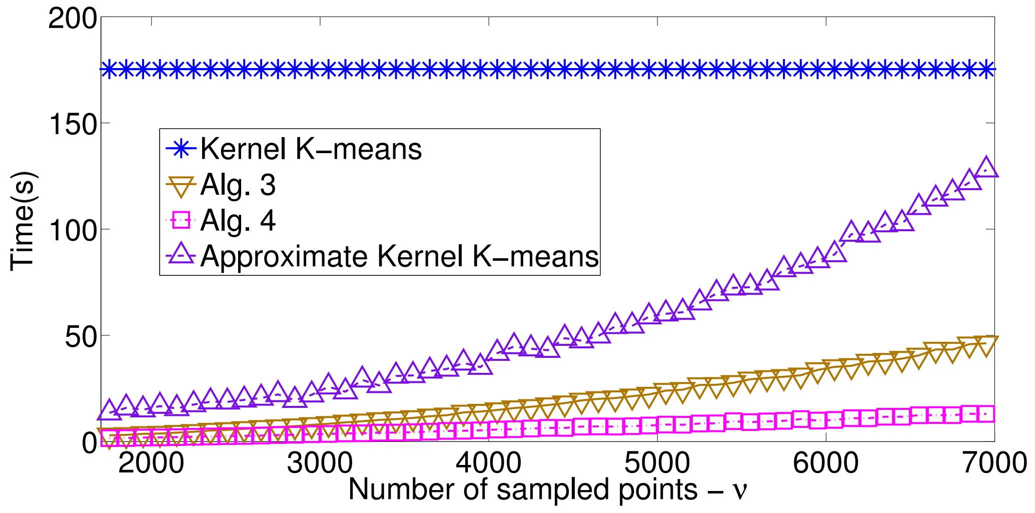


(b) Clustering time (secs)

Figure 2.9: Synthetic data ($D = 5$, $N = 100,000$, $K = 5$).

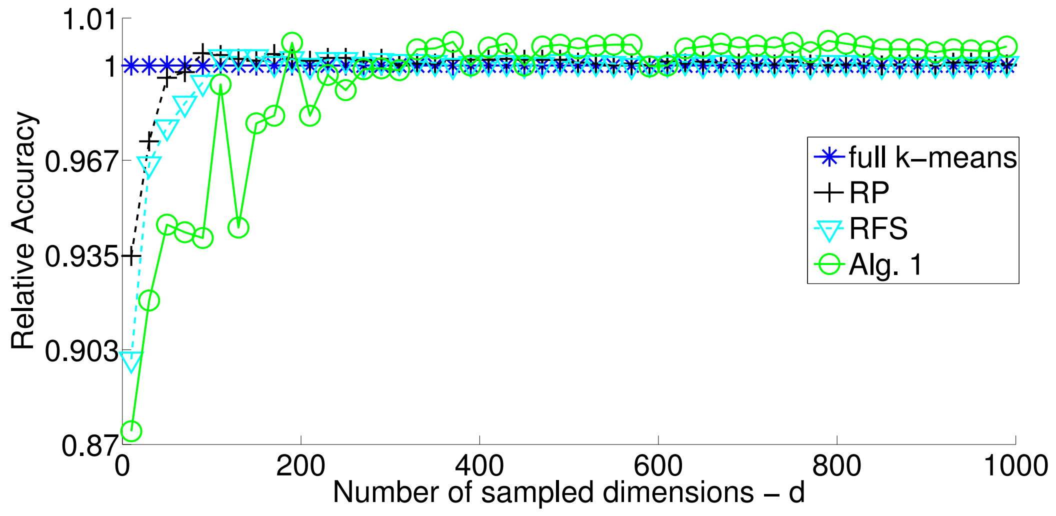


(a) Relative clustering accuracy

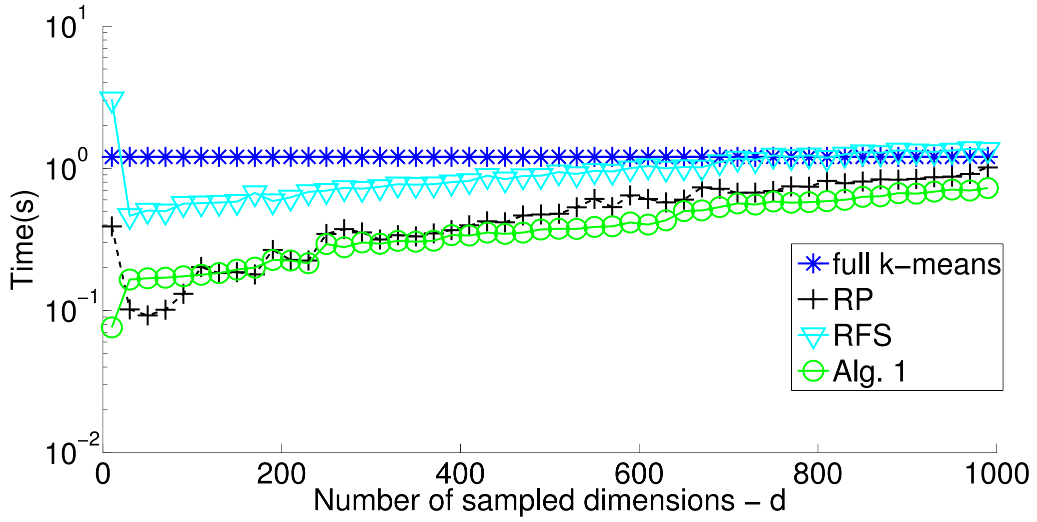


(b) Clustering time (secs)

Figure 2.10: A subset of the MNIST data-set.

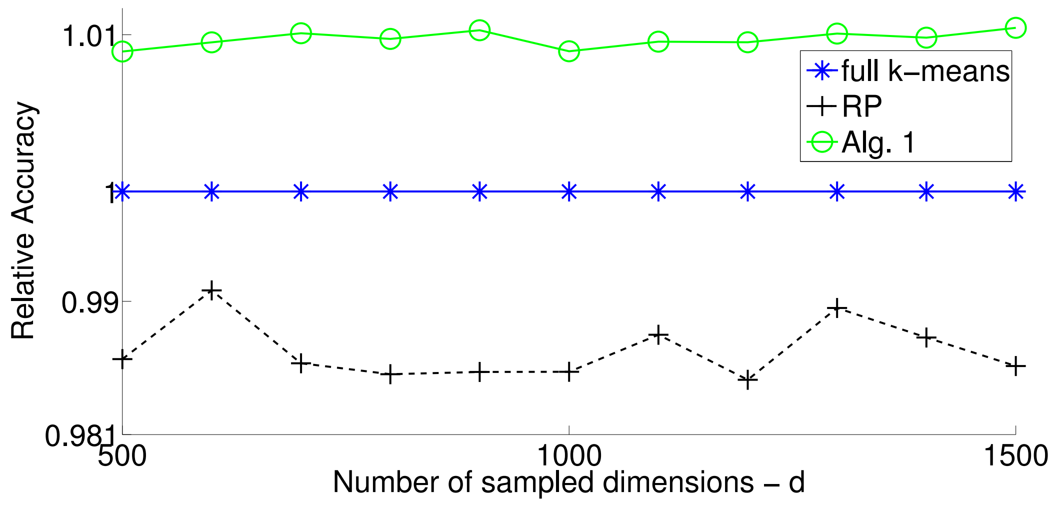


(a) Relative clustering accuracy

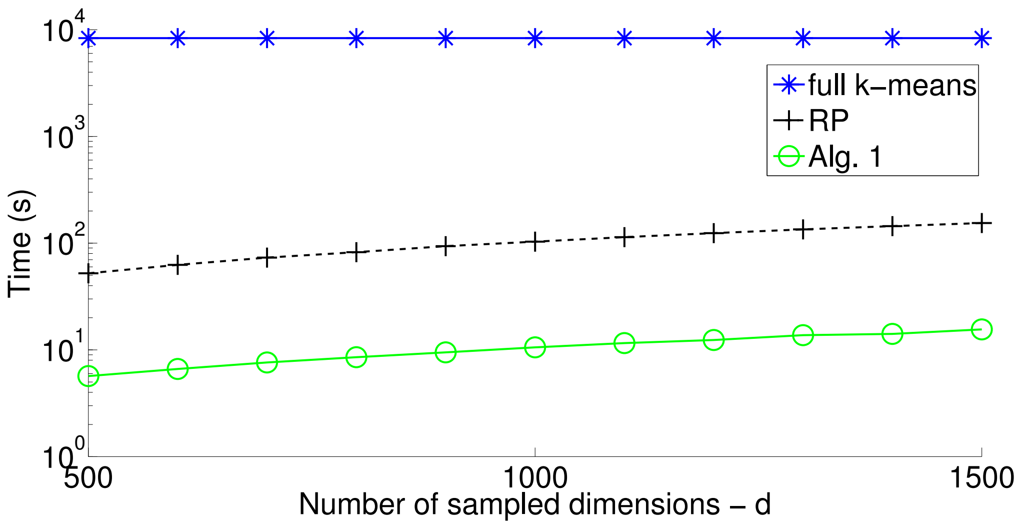


(b) Clustering time (secs)

Figure 2.11: A subset of the ARCENE data-set with multithreading.



(a) Relative clustering accuracy



(b) Clustering time (secs)

Figure 2.12: A subset of the KDDb data-set with multithreading.

Chapter 3

Large-scale Subspace Clustering using SkeVa

The nowadays massive amounts of generated and communicated data present significant challenges in their processing. Subspace clustering (SC) methods, while being able to successfully classify nonlinearly separable datapoints in many occasions, incur prohibitively high computational complexity when processing large-scale data. Inspired by the random sampling and consensus (RANSAC) method, and extending the sketching and validation (SkeVa) scheme introduced in Chapter 2, the present Chapter capitalizes on kernel-based approximation of probability density functions to introduce a randomized scheme for SC, SkeVa-SC, tailored for large-scale data. The proposed scheme exploits also sparsity in the underlying data representation to reduce the computational burden of SC, while achieving high clustering accuracy. Extensive numerical tests on synthetic and real data, as well as a rigorous performance analysis, corroborate the rich potential of the proposed algorithm and the competitive performance relative to state-of-the-art scalable SC approaches.

3.1 Preliminaries

3.1.1 The subspace clustering problem

Consider data $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^N$ drawn from a union of K affine subspaces, each denoted by $\mathcal{S}_k$. Points belonging to a subspace $\mathcal{S}_k$ can be described as:

$$\boldsymbol{x}_i = \mathbf{U}_k \boldsymbol{y}_i + \boldsymbol{\mu}_k + \boldsymbol{e}_i\,, \quad \forall \boldsymbol{x}_i \in \mathcal{S}_k\,, \tag{3.1}$$

where d_k (often $d_k \ll D$) is the dimensionality of the subspace $\mathcal{S}_k$, $\mathbf{U}_k \in \mathbb{R}^{D\times d_k}$ is a basis of $\mathcal{S}_k$, $\boldsymbol{y}_i \in \mathbb{R}^{d_k}$ is the low-dimensional representation of point $\boldsymbol{x}_i$ within $\mathcal{S}_k$, $\boldsymbol{\mu}_k \in \mathbb{R}^D$ is the "centroid" or intercept of $\mathcal{S}_k$, and $\boldsymbol{e}_i \in \mathbb{R}^D$ is the noise vector. The case of $\mathcal{S}_k$ being linear corresponds to $\boldsymbol{\mu}_k = \mathbf{0}$. Using (3.1), *any* point $\boldsymbol{x}_i$ can be described as

$$\boldsymbol{x}_i = \sum_{k=1}^{K} [\boldsymbol{\pi}_i]_k \left(\mathbf{U}_k \boldsymbol{y}_i + \boldsymbol{\mu}_k\right) + \boldsymbol{e}_i\,, \tag{3.2}$$

where $\boldsymbol{\pi}_i$ is known as the cluster assignment vector for point $\boldsymbol{x}_i$ and $[\boldsymbol{\pi}_i]_k$ denotes the kth entry of $\boldsymbol{\pi}_i$ under the constraints of $[\boldsymbol{\pi}_i]_k \geq 0$ and $\sum_{k=1}^{K} [\boldsymbol{\pi}_i]_k = 1$. If $\boldsymbol{\pi}_i \in \{0,1\}^K$ then point $\boldsymbol{x}_i$ can belong to only one subspace (hard clustering), while if $\boldsymbol{\pi}_i \in [0,1]^K$ then point $\boldsymbol{x}_i$ can belong to multiple clusters (soft clustering). In the latter case, $[\boldsymbol{\pi}_i]_k$ can be thought of as the probability that point $\boldsymbol{x}_i$ belongs to cluster $\mathcal{S}_k$.

Given the data matrix $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N] \in \mathbb{R}^{D\times N}$ and the number of subspaces K, the SC task involves extracting the point-to-subspace assignment vectors $\{\boldsymbol{\pi}_i\}_{i=1}^N$, the subspace bases $\{\mathbf{U}_k\}_{k=1}^K$, their dimensions $\{d_k\}_{k=1}^K$, the low-dimensional representations $\{\boldsymbol{y}_i\}_{i=1}^N$, as well as the centroids of the subspaces $\{\boldsymbol{\mu}_k\}_{k=1}^K$ [7]. The SC task can be also cast as the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\Pi},\{\mathbf{U}_k\},\{\boldsymbol{y}_i\},\mathbf{M}} \quad & \sum_{k=1}^{K}\sum_{i=1}^{N} [\boldsymbol{\pi}_i]_k \|\boldsymbol{x}_i - \mathbf{U}_k\boldsymbol{y}_i - \boldsymbol{\mu}_k\|_2^2 \\ \text{subject to (s.to)} \quad & \boldsymbol{\Pi}^\top \mathbf{1} = \mathbf{1} \end{aligned} \tag{3.3}$$

where $\boldsymbol{\Pi} := [\boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_N]$, $\mathbf{M} := [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K]$, and $\mathbf{1}$ denotes the all-ones vector.

The problem in (3.3) is non-convex as $\boldsymbol{\Pi}, \{\mathbf{U}_k\}_{k=1}^K, \{\boldsymbol{y}_i\}_{i=1}^N$ and $\mathbf{M}$ are unknown. However, when the point-to-subspace assignments $\boldsymbol{\Pi}$ and the subspace dimensions $\{d_k\}_{k=1}^K$ are given, bases of the subspaces can be recovered by using the singular value decomposition (SVD) on the data points associated with each subspace. Indeed, given the vectors $\mathbf{X}_k := [\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_{N_k}}] \in \mathbb{R}^{D \times N_k}$, associated with $\mathcal{S}_k$ ($\sum_{k=1}^{K} N_k = N$), a d_k-dimensional subspace basis $\mathbf{U}_k$ can be extracted by taking as $\mathbf{U}_k$ the matrix comprising the first d_k (from the left) column vectors of $\mathbf{U}$, where $\mathbf{X}_k = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ is the SVD of $\mathbf{X}_k - [\boldsymbol{\mu}_k, \ldots, \boldsymbol{\mu}_k]$ and $\boldsymbol{\mu}_k = (1/N_k) \sum_{i \in \mathcal{S}_k} \boldsymbol{x}_i$. On the other hand, when $\{\mathcal{S}_k\}_{k=1}^K$ are given, and in the case of hard clustering, the assignment matrix $\boldsymbol{\Pi}$ can be recovered by finding the closest subspace to each datapoint, i.e., $\forall i \in \{1, 2, \ldots, N\}$,

$$[\boldsymbol{\pi}_i]_k := \begin{cases} 1, & \text{if } k = \arg \min\limits_{k' \in \{1,\ldots,K\}} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_{k'} - \mathbf{U}_{k'} \mathbf{U}_{k'}^\top \boldsymbol{x}_i \right\|_2^2 , \\ 0, & \text{otherwise} , \end{cases} \tag{3.4}$$

where $\|\boldsymbol{x}_i - \boldsymbol{\mu}_k - \mathbf{U}_k \mathbf{U}_k^\top \boldsymbol{x}_i\|_2$ is the distance of point $\boldsymbol{x}_i$ from $\mathcal{S}_k$.

The simple K-subspaces algorithm [45], which is a generalization of the ubiquitous K-means one [46] for SC, builds upon this observation. It solves the SC problem in the following fashion: (i) first, fixing $\boldsymbol{\Pi}$ and then solving for the remaining unknowns; and (ii) fixing $\{\mathcal{S}_k\}_{k=1}^K$ and then solving for $\boldsymbol{\Pi}$.

The previous discussion suggests that due to the employment of SVD, SC entails high computational complexity whenever d_k and/or N_k are massive.

As the proposed algorithm (cf. Section 3.2), will use ideas from Kernel smoothing, a brief primer is provided in Appendix A. Specifically the optimal bandwidth of a kernel estimator, (A.9) will prove useful to attaining good clustering performance.

3.1.2 Prior work

Multiple algorithms have been developed by the machine learning [7] and data-mining community [22], in addition to K-subspaces, to solve (3.3). A probabilistic (soft) counterpart of K-subspaces is the mixture of probabilistic PCA [47], which assumes that data are drawn from a mixture of degenerate Gaussian distributions. Building on the same assumption, the agglomerative lossy compression (ALC) [48] utilizes ideas from rate-distortion theory [49] to minimize the required number of bits to encode each one of the clusters, up to a certain distortion level.

The most successful class of algorithms for solving (3.3) uses spectral clustering [24] to find the point-to-subspace assignments. Algorithms in this class generate first an affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ that captures the similarity between datapoints, and then perform spectral clustering on this matrix. Matrix $\mathbf{A}$ implies a graph $\mathcal{G}$ whose vertices correspond to datapoints and the weights of the edges between datapoints are given by the entries of $\mathbf{A}$. Spectral clustering algorithms form the graph Laplacian matrix:

$$\mathbf{L} := \mathbf{D} - \mathbf{A}\,, \tag{3.5}$$

where $\mathbf{D} \in \mathbb{R}^{N \times N}$ is a diagonal matrix such that (s.t.) $[\mathbf{D}]_{ii} = \sum_{j=1}^{N} [\mathbf{A}]_{ij}$. The algebraic multiplicity of the 0 eigenvalue of $\mathbf{L}$ is equal to the number of connected components in $\mathcal{G}$ [24]. The corresponding eigenvectors of the 0 eigenvalues are indicator vectors of the connected components. In general, $\mathcal{G}$ will be connected however the trailing eigenvectors of $\mathbf{L}$ can still be used to recover cluster assignments as follows: After forming $\mathbf{L}$, spectral clustering algorithms identify the K trailing eigenvectors $\{\mathbf{v}_k \in \mathbb{R}^N\}_{k=1}^{K}$ of $\mathbf{L}$. As the rows(and columns) of $\mathbf{L}$ correspond to datapoints, similarly the rows of $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_K] \in \mathbb{R}^{N \times K}$ also correspond to datapoints. Let $\{\boldsymbol{z}_i \in \mathbb{R}^K\}_{i=1}^{N}$ be the rows of $\mathbf{V}$. This transformation from the vertex space to the space spanned by the trailing K eigenvectors of $\mathbf{L}$ is assumed to enhance and highlight the separability of datapoints. Finally, K-means is employed on $\{\boldsymbol{z}_i \in \mathbb{R}^K\}_{i=1}^{N}$ to recover the point-to-cluster assignments. Normalized versions of spectral clustering [50, 51], where scaled versions of the Laplacian matrix $\mathbf{L}$ are used, have strong relations to graph partitioning problems such as Min-Cut and Ratio-Cut, and can also be employed for subspace clustering.

The sparse subspace clustering algorithm (SSC) [52] exploits the fact that under the union of subspaces model, only a small percentage of datapoints suffice to provide a low-dimensional affine representation of any $\boldsymbol{x}_i$, i.e., $\boldsymbol{x}_i = \sum_{j=1, j \neq i}^{N} w_{ij} \boldsymbol{x}_j$, $\forall i \in \{1, 2, \ldots, N\}$. Specifically, SSC solves the following sparsity-imposing optimization problem:

$$\begin{aligned} \min_{\mathbf{W}} \quad & \|\mathbf{W}\|_1 + \lambda \|\mathbf{X} - \mathbf{X}\mathbf{W}\|_2^2 \\ \text{s.to} \quad & \mathbf{W}\mathbf{1} = \mathbf{1}; \quad \operatorname{diag}(\mathbf{W}) = \mathbf{0} \end{aligned} \tag{3.6}$$

Algorithm 3.1 Unnormalized spectral clustering [24]

```
Input: Data affinity matrix $\mathbf{A}$; number of clusters $K$
Output: Data-cluster associations.
1: Form diagonal matrix $\mathbf{D}$, with entries $[\mathbf{D}]_{ii} = \sum_{j=1}^{N}[\mathbf{A}]_{ij}$
2: $\mathbf{L} = \mathbf{D} - \mathbf{A}$
3: Extract $K$ trailing eigenvectors $\{\mathbf{v}_k \in \mathbb{R}^N\}_{k=1}^K$ of $\mathbf{L}$. Let $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_K] \in \mathbb{R}^{N\times K}$
4: Let $\{\mathbf{z}_i \in \mathbb{R}^K\}_{i=1}^N$ be the rows of $\mathbf{V}$; $\mathbf{z}_i$ corresponds to the $i$-th vertex ($i$-th datapoint).
5: Group $\{\mathbf{z}_i\}_{i=1}^N$ into $k$ clusters $\{C_i\}_{i=1}^k$
```

where $\mathbf{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N] \in \mathbb{R}^{N\times N}$, with $\boldsymbol{w}_i$ being the sparse vector that contains the coefficients for the representation of $\boldsymbol{x}_i$, $\lambda > 0$ is the regularization coefficient, and $\|\mathbf{W}\|_1 := \sum_{i,j=1}^{N}[\mathbf{W}]_{i,j}$. Afterwards, $\mathbf{W}$ is used to create the affinity matrix $[\mathbf{A}]_{ij} := |[\mathbf{W}]_{ij}| + |[\mathbf{W}]_{ji}|$. Finally, spectral clustering is performed using the affinity matrix $\mathbf{A}$ and cluster assignments are identified. Using those cluster assignments, $\mathbf{M}$ is computed by taking sample means per cluster (affine subspace), and $\{\mathbf{U}_k\}_{k=1}^K$, $\{\boldsymbol{y}_i\}_{i=1}^N$ by applying SVD on $\mathbf{X}_k - [\boldsymbol{\mu}_k, \ldots, \boldsymbol{\mu}_k]$. To facilitate the discussion, SSC is summarized in Alg. 3.2.

The low-rank representation algorithm (LRR) [53] works similarly to SSC, but replaces the ℓ_1 norm in (3.6) with the nuclear norm $\|\mathbf{W}\|_* := \sum_{i=1}^{\rho} \sigma_i(\mathbf{W})$, where ρ stands for the rank and $\sigma_i(\mathbf{W})$ for the ith singular value of $\mathbf{W}$. The high clustering accuracy achieved by both SSC and LRR comes at the price of high complexity. Solving (3.6) scales quadratically with the number of datapoints N, on top of performing spectral clustering across K clusters, rendering thus SSC computationally prohibitive for large-scale SC. When data are high-dimensional ($D \gg$), methods based on (statistical) leverage scores, random projections [20], or the more recent sketching and validation (SkeVa) approach, introduced in Chapter 2, can be employed to reduce the dimensionality to a computationally affordable level. When the number of data points is large ($N \gg$), the current state-of-the-art approach, scalable sparse subspace clustering (SSSC) [54], involves drawing randomly $n < N$ data points, performing SSC on them, and expressing the rest of the datapoints according to the clusters identified by that random draw of samples. While this greatly reduces the required complexity, performance can potentially suffer as the random sample might not be representative of the whole dataset, especially in cases where $n \ll N$ under non-equally populated clusters. To alleviate this issue the present thesis introduces a structured trial-and-error approach to identify a "representative" n-point sample from a dataset with $n \ll N$, while maintaining low computational complexity.

Algorithm 3.2 Sparse Subspace Clustering (SSC) [52]

```
Input: Data $\mathbf{X}$; number of clusters $K$; $\lambda$
Output: Data-cluster associations.
1: Solve (3.6) for $\mathbf{W}$.
2: $[\mathbf{A}]_{ij} := |[\mathbf{W}]_{ij}| + |[\mathbf{W}]_{ji}|$.
3: Perform spectral clustering (Alg.3.1) on $\mathbf{A}$.
4: Identify point-to-subspace associations.
```

Regarding kernel smoothing, the majority of developed algorithms address the important issue of bandwidth selection $\mathbf{H}$ of the kernel function to achieve desirable convergence rate properties in the approximation of the unknown pdf [55–58]. The present thesis, however, showcases one of the very few, if not the only one algorithm that provides a framework to randomly choose those kernel functions yielding a small error when estimating a multi-modal pdf.

3.2 Proposed algorithm

Following SkeVa (Chapter 2), the proposed algorithm (named here SkeVa-SC and shown in Alg. 3.3) aims at iteratively finding a representative set of n randomly drawn datapoints, $\check{\mathbf{X}} \in \mathbb{R}^{D\times n}$, run subspace clustering on $\check{\mathbf{X}}$, and associate the remaining datapoints $\tilde{\mathbf{X}} = \mathbf{X} \setminus \check{\mathbf{X}} \in \mathbb{R}^{D\times N-n}$ with the subspaces extracted from $\check{\mathbf{X}}$. SkeVa-SC is performed for a prescribed number of $R_{\max}$ iterations, where each iteration consists of two stages: A random-sketching and a validation phase. The structure and philosophy of this algorithm is similar to the DiSkeVa (cf. Alg. 4 Chapter 2) algorithm that has been used successfully for clustering high-dimensional data.

At each iteration (indexed here by r), the random-sketching stage of the algorithm is performed as follows: A randomly chosen population of the data is drawn, $\check{\mathbf{X}}^{(r)}$, and a pdf $\check{p}^{(r)}(\boldsymbol{x})$, induced by this random sample, is estimated. As the clusters are assumed to be well-separated, $\check{p}^{(r)}(\boldsymbol{x})$ is expected to be multimodal. Thus, $\check{p}^{(r)}(\boldsymbol{x})$ is compared with a unimodal pdf $\check{p}_0^{(r)}(\boldsymbol{x})$, using a measure $d(\check{p}^{(r)}, \check{p}_0^{(r)})$ of pdf discrepancy. If $\check{p}^{(r)}$ is sufficiently different from $\check{p}_0^{(r)}$, i.e., $d(\check{p}^{(r)}, \check{p}_0^{(r)}) \geq \Delta_0$, where Δ_0 is some pre-defined threshold, SkeVa-SC proceeds to the validation stage; otherwise, SkeVa-SC advances to the next iteration $r+1$, without performing the validation one.

At the validation stage of SkeVa-SC, another random sample of n' datapoints, different from the one in $\check{\mathbf{X}}^{(r)}$, is drawn, forming $\bar{\mathbf{X}}^{(r)} \in \mathbb{R}^{D\times n'}$. The purpose of this stage is to evaluate how well $\check{\mathbf{X}}$ represents the whole dataset. The pdf $\bar{p}^{(r)}(\boldsymbol{x})$ of $\bar{\mathbf{X}}^{(r)}$ is estimated and compared to $\check{p}^{(r)}(\boldsymbol{x})$ using $d(\check{p}^{(r)}, \bar{p}^{(r)})$. A score $\psi[d(\check{p}^{(r)}, \bar{p}^{(r)})]$ is assigned to $\check{\mathbf{X}}^{(r)}$, using a non-decreasing scoring function $\psi : \mathbb{R} \to \mathbb{R}$.

Finally, after $R_{\max}$ iterations, the set of n datapoints $\check{\mathbf{X}}^{(r^*)}$ that yielded the highest score, $r^* := \arg\max_r \psi[d(\check{p}^{(r)}, \bar{p}^{(r)})]$ is selected and SC (SSC, or any other algorithm) is performed on it, i.e.,

$$\begin{aligned} \min_{\mathbf{W}} \quad & \|\mathbf{W}\|_1 + \lambda \left\| \check{\mathbf{X}}^{(r^*)} - \check{\mathbf{X}}^{(r^*)}\mathbf{W} \right\|_2^2 \\ \text{s.to} \quad & \mathbf{W}\mathbf{1} = \mathbf{1}; \quad \operatorname{diag}(\mathbf{W}) = \mathbf{0}\,. \end{aligned} \tag{3.7}$$

The remaining datapoints $\bar{\mathbf{X}}^{(r^*)} := \mathbf{X} \setminus \check{\mathbf{X}}^{(r^*)}$ are associated with the clusters defined by $\check{\mathbf{X}}^{(r^*)}$. This association can be performed either by using the residual minimization method, described in SSSC [54], or, if subspace dimensions are known, by identifying the subspace that is closest to each datapoint, as in (3.4).

Remark 7. *Threshold Δ_0 can be updated across iterations. If $\Delta_0^{(0)} = -\infty$ stands for the initial threshold value, $\Delta_0^{(r)}$, at iteration $r \in \{1, \ldots, R_{\max}\}$, can be updated as follows:*

$$\Delta_0^{(r)} := \begin{cases} d(\check{p}^{(r)}, \check{p}_0^{(r)})\,, & \textit{if } d(\check{p}^{(r)}, \check{p}_0^{(r)}) \geq \Delta_0^{(r-1)}\,, \\ \Delta_0^{(r-1)}\,, & \textit{otherwise}\,. \end{cases} \tag{3.8}$$

Remark 8. *As the iterations of SkeVa-SC are independent from each other, they can be readily parallelized, using schemes such as MapReduce [34].*

To estimate the densities at each step of SkeVa-SC, kernel density estimators, or kernel smoothers (cf. Appendix A) are employed. In other words, SkeVa-SC, much like DiSkeVa, (cf. Alg. 4 Chapter 2) attempts to solve the following optimization problem: $\min_{\check{\mathbf{X}}^{(r)}} d(f, \hat{f})$, where $\hat{f}(\boldsymbol{x}) := (1/n)\sum_{i=1}^{n} K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{x}_i^{(r)})$, with $\boldsymbol{x}_i^{(r)}$ being the ith column of $\check{\mathbf{X}}^{(r)}$. As the pdf f to be estimated is generally unknown, a sensible choice for the bandwidth matrix is $\mathbf{H} := h^2\mathbf{I}_D$, $h > 0$, as it provides isotropic smoothing across all dimensions and greatly simplifies computation.

To perform computations in closed form, the dissimilarity $d(\cdot,\cdot)$ is chosen as the integrated square error (ISE):

$$d_{\mathrm{ISE}}(f, g) := \int (f(\boldsymbol{x}) - g(\boldsymbol{x}))^2 \, d\boldsymbol{x} \,, \tag{3.9}$$

or, as in Chapter 2, the Cauchy-Schwarz divergence [33]:

$$d_{\mathrm{CS}}(f, g) := -\log \frac{\left(\int f(\boldsymbol{x})g(\boldsymbol{x})d\boldsymbol{x}\right)^2}{\int f^2(\boldsymbol{x})d\boldsymbol{x} \int g^2(\boldsymbol{x})d\boldsymbol{x}} \,. \tag{3.10}$$

Moreover, the kernel that will be used is the following Gaussian multivariate function $K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{x}_i) = \phi_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{x}_i)$, where $\phi_{\mathbf{H}}$ is defined in (A.7).

The estimated pdfs that are used by SkeVa-SC are defined as follows:

$$\hat{f}^{(r)}(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n} \phi_{\mathbf{H}}\left(\boldsymbol{x} - \boldsymbol{x}_i^{(r)}\right) \,, \tag{3.11a}$$

$$\hat{f}_0^{(r)}(\boldsymbol{x}) := \phi_{\mathbf{H}_0}\left(\boldsymbol{x} - \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i^{(r)}\right) \,, \tag{3.11b}$$

$$\tilde{f}^{(r)}(\boldsymbol{x}) = \frac{1}{n'}\sum_{i=1}^{n'} \phi_{\mathbf{H}'}\left(\boldsymbol{x} - \tilde{\boldsymbol{x}}_i^{(r)}\right) \,, \tag{3.11c}$$

where $\bar{\boldsymbol{x}}_i^{(r)}$ is the ith column of $\bar{\mathbf{X}}^{(r)}$, and $\mathbf{H}, \mathbf{H}_0, \mathbf{H}'$ are appropriately defined bandwidth matrices. It is then easy to show that for a measure such as (3.9) or (3.10), and by using property (A.8) several of the dissimilarities of pdfs that were previously mentioned can be

Algorithm 3.3 Sketching and validation SC (SkeVa-SC)

Input: Data $\mathbf{X}$; maximum number of iterations $R_{\max}$; bandwidth h

Output: Clustered data; bases of subspaces

```
1: for $r = 1$ to $R_{\max}$ do
2:     Sample $n \ll N$ datapoints of $\mathbf{X}$; call sampled matrix $\check{\mathbf{X}}^{(r)}$.
3:     Estimate pdf $\check{p}^{(r)}$ of $\check{\mathbf{X}}^{(r)}$; evaluate dissimilarity $d(\check{p}^{(r)}, \check{p}_0^{(r)})$ from a unimodal pdf
       $\check{p}_0^{(r)}$ [cf. (3.11)].
4:     if $d(\check{p}^{(r)}, \check{p}_0^{(r)}) \geq \Delta$ then
5:         Sample $n' \ll N$ new datapoints $\bar{\mathbf{X}}^{(r)}$; evaluate pdf $\bar{p}^{(r)}$ of $\bar{\mathbf{X}}^{(r)}$.
6:         Evaluate dissimilarity $d(\check{p}^{(r)}, \bar{p}^{(r)})$ and assign a score $\psi[d(\check{p}^{(r)}, \bar{p}^{(r)})]$ to $\check{\mathbf{X}}^{(r)}$.
7:     end if
8: end for
9: $r^* := \arg\max_r \psi[d(\check{p}^{(r)}, \bar{p}^{(r)})]$.
10: Perform SSC (Alg. 3.2) on $\check{\mathbf{X}}^{(r^*)}$; obtain data-cluster associations.
11: Associate $\mathbf{X} \setminus \check{\boldsymbol{X}}^{(r^*)}$ to clusters defined in step 10.
```

computed in closed-form as follows:

$$\begin{aligned} d_{\text{ISE}}(\check{p}^{(r)}, \check{p}_0^{(r)}) = & \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \phi_{2\mathbf{H}}(\boldsymbol{x}_i^{(r)} - \boldsymbol{x}_j^{(r)}) \\ & + \phi_{2\mathbf{H}_0}(\mathbf{0}) \\ & - \frac{2}{n} \sum_{i=1}^{n} \phi_{\mathbf{H}+\mathbf{H}_0} \left(\boldsymbol{x}_i^{(r)} - \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i^{(r)} \right) . \end{aligned} \tag{3.12a}$$

$$\begin{aligned} d_{\text{ISE}}(\check{p}^{(r)}, \bar{p}^{(r)}) = & \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \phi_{2\mathbf{H}}(\boldsymbol{x}_i^{(r)} - \boldsymbol{x}_j^{(r)}) \\ & + \frac{1}{n'^2} \sum_{i=1}^{n'} \sum_{j=1}^{n'} \phi_{2\mathbf{H}'}(\bar{\boldsymbol{x}}_i^{(r)} - \bar{\boldsymbol{x}}_j^{(r)}) \\ & - \frac{2}{nn'} \sum_{i=1}^{n} \sum_{j=1}^{n'} \phi_{\mathbf{H}+\mathbf{H}'}(\boldsymbol{x}_i^{(r)} - \bar{\boldsymbol{x}}_j^{(r)}) . \end{aligned} \tag{3.12b}$$

$$\begin{aligned}
d_{\mathrm{CS}}(\check{p}^{(r)}, \check{p}_0^{(r)}) = & -2\log\left[\frac{1}{n}\sum_{i=1}^{n}\phi_{\mathbf{H}+\mathbf{H}_0}\left(\boldsymbol{x}_i^{(r)} - \frac{1}{n}\sum_{j=1}^{n}\boldsymbol{x}_i^{(r)}\right)\right] \\
& + \log\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\phi_{2\mathbf{H}}\left(\boldsymbol{x}_i^{(r)} - \boldsymbol{x}_j^{(r)}\right) \\
& + \log\phi_{2\mathbf{H}_0}(\mathbf{0})\,. \qquad (3.13a) \\
d_{\mathrm{CS}}(\check{p}^{(r)}, \bar{p}^{(r)}) = & -2\log\left[\frac{1}{nn'}\sum_{i=1}^{n}\sum_{j=1}^{n'}\phi_{\mathbf{H}+\mathbf{H}'}\left(\boldsymbol{x}_i^{(r)} - \bar{\boldsymbol{x}}_j^{(r)}\right)\right] \\
& + \log\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\phi_{2\mathbf{H}}\left(\boldsymbol{x}_i^{(r)} - \boldsymbol{x}_j^{(r)}\right) \\
& + \log\frac{1}{n'^2}\sum_{i=1}^{n'}\sum_{j=1}^{n'}\phi_{2\mathbf{H}'}\left(\bar{\boldsymbol{x}}_i^{(r)} - \bar{\boldsymbol{x}}_j^{(r)}\right)\,. \qquad (3.13b)
\end{aligned}$$

Given that the computation of the Gaussian dissimilarities between n D-dimensional datapoints incurs a complexity of $\mathcal{O}(Dn^2)$, Alg. 3.3 yields a computational complexity of $o[DR_{\max}(n^2 + nn' + n'^2)]$ per iteration, if the Gaussian kernel is utilized. As with any algorithm that involves kernel smoothing, the choice of bandwidth matrices $\mathbf{H}, \mathbf{H}', \mathbf{H}_0$ is critical to the performance of Alg. 3.3.

3.3 Performance Analysis

The crux of Alg. 4.1 is the $R_{\max}$ number of independent random draws of n data to identify a subset $\check{\mathbf{X}}^{(r^*)}$ that "represents well" the whole population of data. Since SkeVa-SC aims at large-scale clustering ($N \gg n$), it is natural to ask whether $R_{\max}$ iterations suffice to mine a subset of data whose pdf approximates well the unknown f. It becomes therefore crucial, prior to any implementation of SkeVa-SC, to have an estimate of the minimum number of $R_{\max}$ that ensures a "good draw" with high probability. Such concerns are not taken into account in SSSC [54], where only a *single* draw is performed prior to applying SC. Along the previous lines, this section provides with a theoretical analysis which establishes such a lower-bound on $R_{\max}$, when the underlying pdf f abides by GMM. Due to the universal approximation properties of GMM [5,59,60], such a generic assumption on f is also met in the mixture of probabilistic PCA [47] as well as ALC [48].

Performance analysis will be based on the following generic assumption. It pertains to modeling the multimodal pdf f of the data by a mixture of Gaussian ones. This assumption seems appropriate as any pdf or multivariate function that is t-th order integrable $t \in [0, \infty)$, can be approximated by a mixture of appropriately many Gaussians [31,59].

Assumption 1. *Data are generated according to the GMM model, i.e., the pdf f of the data is given by*

$$f(\boldsymbol{x}) = \sum_{\ell=1}^{L} w_\ell \phi_{\boldsymbol{\Sigma}_\ell}(\boldsymbol{x} - \boldsymbol{\mu}_\ell), \quad \sum_{\ell=1}^{L} w_\ell = 1\,, \tag{3.14}$$

where $L \geq K$, $\boldsymbol{\mu}_\ell \in \mathbb{R}^D$ and $\boldsymbol{\Sigma}_\ell \in \mathbb{R}^{D\times D}$ stand for the mean vector and the covariance matrix of the ℓth Gaussian pdf, respectively, and $\{w_\ell\}_{l=1}^{L} \subset [0,1]$ are the mixing coefficients.

Using As. 1, the mean of the entire dataset is $\boldsymbol{\mu}_0 = \sum_{\ell=1}^{L} w_\ell \boldsymbol{\mu}_\ell$, and can be estimated by the sample mean of all data drawn from f.

Definition 1. *A "dissimilarity" function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a metric or a distance if the following properties hold $\forall f_1, f_2, f_3 \in \mathcal{X}$: 1. $d(f_1, f_2) \geq 0$; 2. $d(f_1, f_2) = 0 \Leftrightarrow f_1 = f_2$; 3. $d(f_1, f_2) = d(f_2,$ Property 4 (depicted in Fig. 3.1) is widely known as the triangle inequality one. A semi-distance is a function d for which properties 1, 3, 4, and $[d(f, f) = 0, \forall f \in \mathcal{X}]$ hold. A divergence is a function d where $\mathcal{X}$ is the space of pdfs, and for which only properties 1 and 2 hold. The class of Bregman divergences are generalizations of the squared Euclidean distance and include the Kullback-Leibler [61] as well as the Itakura-Saito one [62], among others. Furthermore, generalized symmetric Bregman divergences, such as the Jensen-Bregman one, satisfy the triangle inequality property [63]. Although d_{ISE} is not a distance, since it does not satisfy the triangle inequality property, $\sqrt{d_{ISE}}$ (the ℓ_2-norm) qualifies as a distance function.*

If $\hat{f}$ denotes an estimate of the data pdf f, and f_0 stands for a reference pdf (a rigorous definition will follow), Fig. 3.1 depicts $f, \hat{f}$ and f_0 as points in the statistical manifold $\mathcal{X}$ [64], namely the space of probability distributions. Letting $\delta' := d(f, f_0)$, the triangle inequality property suggests that

$$|d(f,\hat{f}) - \delta'| \leq d(f_0,\hat{f}) \leq d(f,\hat{f}) + \delta' \,. \tag{3.15}$$

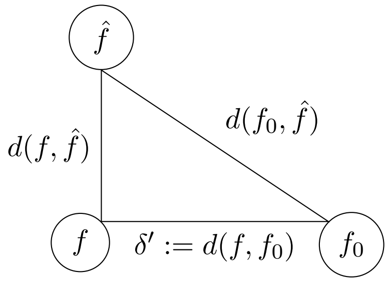


Figure 3.1: The triangle inequality property in the statistical manifold.

Definition 2. *Given a dissimilarity function d which satisfies the triangle inequality property in Def. 1, an event at an iteration of Alg. 4.1, denoted as $\mathcal{B}_\delta$, is deemed "bad" if the dissimilarity between the kernel-based estimator $\hat{f}$ and the true pdf f is larger than or equal to some prescribed value δ, i.e.,*

$$\mathcal{B}_\delta := \{d(\hat{f},f) \geq \delta\} \,, \tag{3.16}$$

where $\{St(\ldots)\}$ denotes the event of the statement $St(\ldots)$ being true. Naturally, an $\hat{f}$ such that the complement $\mathcal{B}_\delta^{\complement}$ of $\mathcal{B}_\delta$ holds true will be called a "good" estimate of the underlying f. Given $f_0 \in \mathcal{X}$, and thus $\delta' := d(f,f_0)$, the triangle inequality property suggests that $\delta' - d(f_0,\hat{f}) \leq d(f,\hat{f})$. This implies that for a fixed δ', the smaller $d(f_0,\hat{f})$ is, the larger $d(f,\hat{f})$ becomes. For any arbitrarily fixed $\delta_0 \leq \delta' - \delta$, any $\hat{f}$ for which $d(f_0,\hat{f}) < \delta_0$ implies that $\delta \leq \delta' - \delta_0 < \delta' - d(f_0,\hat{f}) \leq d(f,\hat{f})$, i.e., $\mathcal{B}_\delta$ occurs. In other words,

$$\forall \delta_0 \in (0, d(f,f_0) - \delta], \quad \{d(f_0,\hat{f}) < \delta_0\} \subseteq \mathcal{B}_\delta \,, \tag{3.17}$$

Here, f_0 is chosen as the unimodal Gaussian pdf $f_0(\boldsymbol{x}) := \phi_{\mathbf{H}_0}(\boldsymbol{x} - \boldsymbol{\mu}_0)$ that is centered around $\boldsymbol{\mu}_0$. This is in contrast with the multimodal nature of the true f, under the assumption of a number $K > 1$ of well-separated clusters. According to (3.17), an estimate $\hat{f}$ that is "close" to the unimodal f_0 will trigger the "bad" event $\mathcal{B}_\delta$.

Remark 9. *Note that, in this section, f_0 is centered in the sample mean of the whole dataset, while in Alg. 4.1, $\hat{f}_0$ at each iteration is centered around the sample mean of the sampled data. This is to avoid a step that incurs $\mathcal{O}(N)$ complexity in SkeVa-SC. If the dataset mean is provided (via a preprocessing step) then the unimodal pdf at each iteration $\hat{f}_0^{(r)}$ can be replaced by the pdf induced by the dataset mean f_0.*

The maximum required number of iterations $R_{\max}$ can be now lower-bounded in a manner similar to RANSAC-type algorithms [25].

Theorem 1.

1. *Given a distance function d [cf. Def. 2], a $\delta > 0$, a "success" probability $p \in (0,1)$ of Alg. 4.1, i.e., the probability that after $R_{\max}$ iterations a random draw of data-points yields an estimate $\hat{f}$ that satisfies $\mathcal{B}_\delta^{\complement}$ [cf. (3.16)], Alg. 4.1 requires*

$$R_{\max} \geq \frac{\log(1-p)}{\log\left(1 - \frac{\mathbb{E}\left[d(\hat{f}, f_0)\right]}{d(f, f_0) - \delta}\right)}, \tag{3.18}$$

   iterations to succeed in identifying a "good" $\hat{f}$, where expectation is taken w.r.t. the data pdf f, i.e.,

$$\mathbb{E}\left[d(\hat{f}, f_0)\right] := \int d(\hat{f}, f_0) f(\boldsymbol{x}) d\boldsymbol{x}\,.$$

2. *Given a probability $q \in (0,1)$ and under As. 1 as well as $d := \sqrt{d_{ISE}}$ [cf. (3.9)], the following lower bound holds with probability $1 - q$:*

$$R_{\max} \geq \frac{\log(1-p)}{\log\left(1 - \frac{\zeta}{\theta_1 + \theta_2}\right)}\,. \tag{3.19}$$

Here,

$$\begin{aligned}\zeta^2 := {} & \mathbb{E}[d_{ISE}(\hat{f}, f_0)] \\ = {} & \frac{1}{(4\pi)^{D/2}|\mathbf{H}_0|^{1/2}} + \frac{1}{n}\frac{1}{(4\pi)^{D/2}|\mathbf{H}|^{1/2}} \\ & + \left(1 - \frac{1}{n}\right)\mathbf{w}^\top \boldsymbol{\Omega}_2 \mathbf{w} \\ & - 2\sum_\ell w_\ell \phi_{\mathbf{H}+\mathbf{H}_0+\boldsymbol{\Sigma}_\ell}(\boldsymbol{\mu}_\ell - \boldsymbol{\mu}_0)\,, \end{aligned} \tag{3.20a}$$

$$\begin{aligned}\theta_1 := {} & \left[-\frac{2\log(q/2)}{nh\,(4\pi)^{D/2}}\right]^{1/2} + \left\{\frac{1}{n(4\pi)^{D/2}|\mathbf{H}|^{1/2}}\right. \\ & \left. + \mathbf{w}^\top\left[\left(1-\frac{1}{n}\right)\boldsymbol{\Omega}_2 - 2\boldsymbol{\Omega}_1 + \boldsymbol{\Omega}_0\right]\mathbf{w}\right\}^{1/2}\,, \end{aligned} \tag{3.20b}$$

$$\begin{aligned}\theta_2 := {} & \left\{\mathbf{w}^\top\boldsymbol{\Omega}_0\mathbf{w} + \frac{1}{(4\pi)^{D/2}|\mathbf{H}_0|^{1/2}}\right. \\ & \left. -2\sum_\ell w_\ell \phi_{\boldsymbol{\Sigma}_\ell+\mathbf{H}_0}(\boldsymbol{\mu}_\ell - \boldsymbol{\mu}_\mathbf{0})\right\}^{1/2}\,, \end{aligned} \tag{3.20c}$$

where $\mathbf{w} := [w_1, w_2, \dots, w_L]^\top$ *is a vector containing the mixing coefficients of (3.14) and* $\boldsymbol{\Omega}_\alpha \in \mathbb{R}^{L\times L}, \alpha \in \{0,1,2\}$*, is a matrix whose* (i,j)*th entry is given by*

$$[\boldsymbol{\Omega}_\alpha]_{ij} = \phi_{\alpha\mathbf{H}+\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_j}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)\,. \tag{3.21}$$

Proof. By definition, $1-p$ is the probability that Alg. 4.1 fails to identify a "good" draw (cf. Def. 2) after $R_{\max}$ iterations. Since iterations are independent it holds that

$$[\Pr(\mathcal{B}_\delta)]^{R_{\max}} = 1 - p\,. \tag{3.22}$$

Then, $R_{\max}$ can be lower-bounded as

$$\begin{aligned}[\Pr(\mathcal{B}_\delta)]^{R_{\max}} \le 1 - p \Leftrightarrow {} & R_{\max}\log\left(\Pr(\mathcal{B}_\delta)\right) \le \log(1-p) \\ \Leftrightarrow {} & R_{\max} \ge \frac{\log(1-p)}{\log\left(\Pr(\mathcal{B}_\delta)\right)}\,, \end{aligned} \tag{3.23}$$

where the last inequality follows from the trivial fact that $\Pr(\mathcal{B}_\delta) < 1 \Leftrightarrow \log\Pr(\mathcal{B}_\delta) < 0$.

Using (3.17), $\Pr(\mathcal{B}_\delta)$ is lower-bounded as

$$\begin{aligned}\Pr(\mathcal{B}_\delta) &= \Pr\left(d(\hat{f}, f) \ge \delta\right) \\ &\ge \Pr\left(d(\hat{f}, f_0) < \delta_0\right)\,, \end{aligned} \tag{3.24}$$

where $\delta_0 := d(f, f_0) - \delta$; thus when δ is fixed, so is δ_0. Furthermore, by Markov's inequality,

$$\begin{aligned}\Pr\left(d(\hat{f}, f_0) < \delta_0\right) &= 1 - \Pr\left(d(\hat{f}, f_0) \geq \delta_0\right) \\ &\geq 1 - \frac{\mathbb{E}\left[d(\hat{f}, f_0)\right]}{\delta_0}. \end{aligned} \tag{3.25}$$

Combining (3.24) with (3.25) yields

$$\Pr(\mathcal{B}_\delta) \geq 1 - \frac{\mathbb{E}\left[d(\hat{f}, f_0)\right]}{\delta_0}. \tag{3.26}$$

Using now (3.26) and that $\log(1-p) < 0$, it is easy to establish (3.18) via (3.23). In the case where $1 - \mathbb{E}[d(\hat{f}, f_0)]/\delta_0 \leq 0$, (3.26) is uninformative, and 1 is used as the trivial lower bound of $R_{\max}$. This completes the proof of Thm. 1.1.

Regarding the proof of Thm. 1.2, closed-form expressions of $\mathbb{E}[d(\hat{f}, f_0)]$ as well as values for δ, δ_0 and δ', using $d_{\text{ISE}}(\cdot, \cdot)$, to establish (3.19) are provided in Appendix C. $\square$

Fig. 3.2 depicts the $R_{\max}$ values as the number of sampled data n increases for a synthetic one-dimensional ($D = 1$) dataset of $N = 480$ with $K = 3$ clusters generated according to (3.14). The cluster means are $\{-1, 1, 2\}$, the variances are $\{0.5, 0.5, 0.5\}$, and the number of data per cluster are $\{100, 180, 200\}$, respectively. The $R_{\max}$ values are obtained using $(p, q) := (0.99, 0.01)$ in Thm. 1.2. The results shown in Fig. 3.2 are intuitively pleasing: As the number of sampled points n increases, the required number of random draws decreases, and at $n = 480$ only one draw is required.

For the same dataset, Fig. 3.3 depicts the accuracy [% of correctly clustered data (cf. Section 3.4)] of Alg. 4.1 using K-means clustering instead of SSC, as $R_{\max}$ increases. Here, the number of sampled data n is set to $n = 10$ and the number of data for the validation phase is set to $n' = 50$. Alg. 4.1 is compared to the simple scheme of taking a *single* random draw of n data and performing K-means. The vertical red line in Fig. 3.3 indicates the value of $R_{\max}$ provided by Thm. 1.2. The results are averaged over 10 independent Monte Carlo runs.

When reliable estimates of f are unknown, values for $R_{\max}$ can be estimated on-the-fly by using (3.18) and sample averaging. The ensemble averages of Thm. 1.1 can be replaced

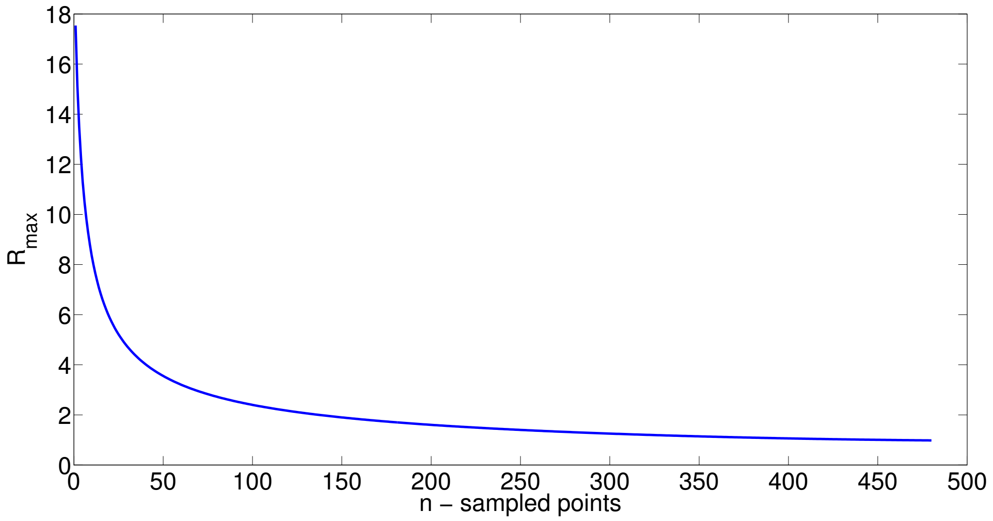


Figure 3.2: $R_{\max}$ values versus number of sampled points n for $K = 3$ clusters generated by (3.14) with $D = 1$, cluster means $\{-1, 1, 2\}$, variances $\{0.5, 0.5, 0.5\}$ and number of points per cluster $\{100, 180, 200\}$.

by sample ones, where averaging takes place across iterations. In particular, $\mathbb{E}[d(\hat{f}, f_0)]$ [cf. (3.18)] is replaced by $\bar{d}^{(r)}(\hat{f}, f_0)$, which denotes the sample average of $d(\hat{f}, f_0)$, as of iteration r. Moreover, since the number n' of data used in the validation phase is set to be larger than or equal to the number n of data drawn in the sketching phase of Alg. 4.1, $\tilde{f}$ of (3.11c) provides potentially better estimates of f than $\hat{f}$ does. As a result, distance $d(\tilde{f}^{(i)}, f_0^{(i)})$ stands now as a surrogate to δ' [cf. (3.15)]. Further, by $d := \sqrt{d_{\mathrm{ISE}}}$ and by using (C.3), (C.14), as well as replacing ensemble averages by sample ones, an estimate of δ_0 at iteration r of Alg. 4.1 is

$$\bar{\delta}_0^{(r)} := \sqrt{-\frac{2\log(q/2)}{nh\,(4\pi)^{D/2}}} + \bar{d}^{(r)}(\tilde{f}, \hat{f}) + \bar{d}^{(r)}(\tilde{f}, f_0)\,, \tag{3.27}$$

where $\bar{d}^{(r)}(\tilde{f}, \hat{f})$ and $\bar{d}^{(r)}(\tilde{f}, f_0)$ are the sample averages of $d(\tilde{f}, \hat{f})$ and $d(\tilde{f}, f_0)$, respectively. Therefore, according to (3.11), the following quantities can be computed at each iteration of Alg. 4.1:

$$\bar{d}^{(r)}(f_0, \hat{f}) = \frac{1}{r}\sum_{i=1}^{r} d(f_0^{(i)}, \hat{f}^{(i)})\,, \tag{3.28a}$$

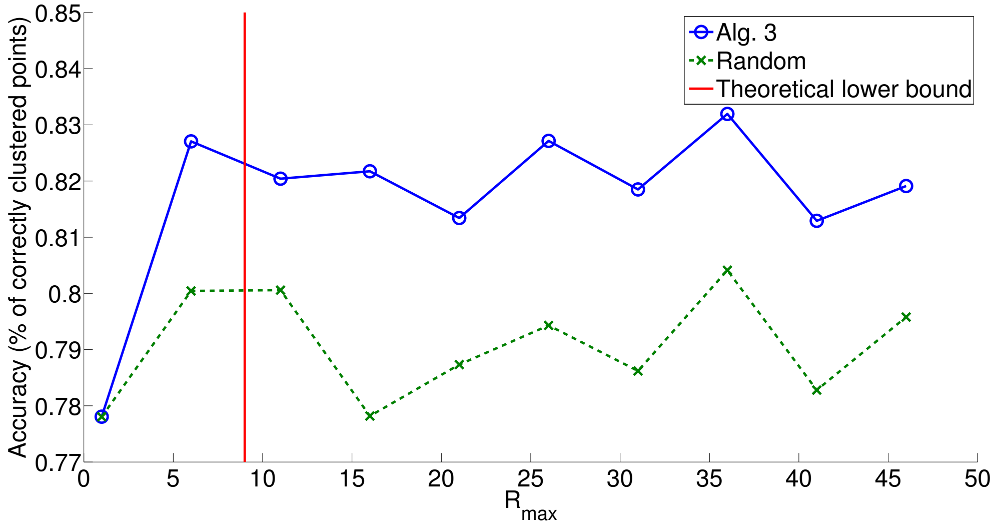


Figure 3.3: Accuracy (% of correctly clustered data) versus $R_{\max}$ number of sampled points n for the data used in Fig. 3.2. Alg. 4.1 is compared to the simple scheme of taking a single random draw of n data, followed by clustering. The vertical line represents the $R_{\max}$ value given by Theorem 1.

$$\bar{d}^{(r)}(\tilde{f}, \hat{f}) = \frac{1}{r}\sum_{i=1}^{r} d(\tilde{f}^{(i)}, \hat{f}^{(i)}), \tag{3.28b}$$

$$\bar{d}^{(r)}(\tilde{f}, f_0) = \frac{1}{r}\sum_{i=1}^{r} d(\tilde{f}^{(i)}, f_0^{(i)}). \tag{3.28c}$$

Consequently, (3.18) can be approximated at each iteration r using equations (3.27) and (3.28a)-(3.28c) as

$$\hat{R}_{\max}^{(r)} \geq \frac{\log(1-p)}{\log\left(1 - \frac{\bar{d}^{(r)}(f_0,\hat{f})}{\bar{\delta}_0^{(r)}}\right)} =: \check{R}_{\max}^{(r)}. \tag{3.29}$$

Based on the previous sample averages, a simple rule for updating $\hat{R}_{\max}^{(r)}$, which quantifies the maximum number of iterations to be executed in Alg. 4.1, as of iteration r, is

$$\hat{R}_{\max}^{(r)} := \max\left\{\check{R}_{\max}^{(r)}, R_0\right\}, \tag{3.30}$$

where R_0 is a prescribed absolute minimum number of iterations. Fig. 3.4 shows the values of (3.30) as iterations of Alg. 4.1 are run for the dataset used in Figs. 3.2 and 3.3. Here

$R_0 := 3$, and $\hat{R}_{\max}^{(r)}$ approaches the theoretical value given by Thm. 1, as values of the distances are averaged across iterations.

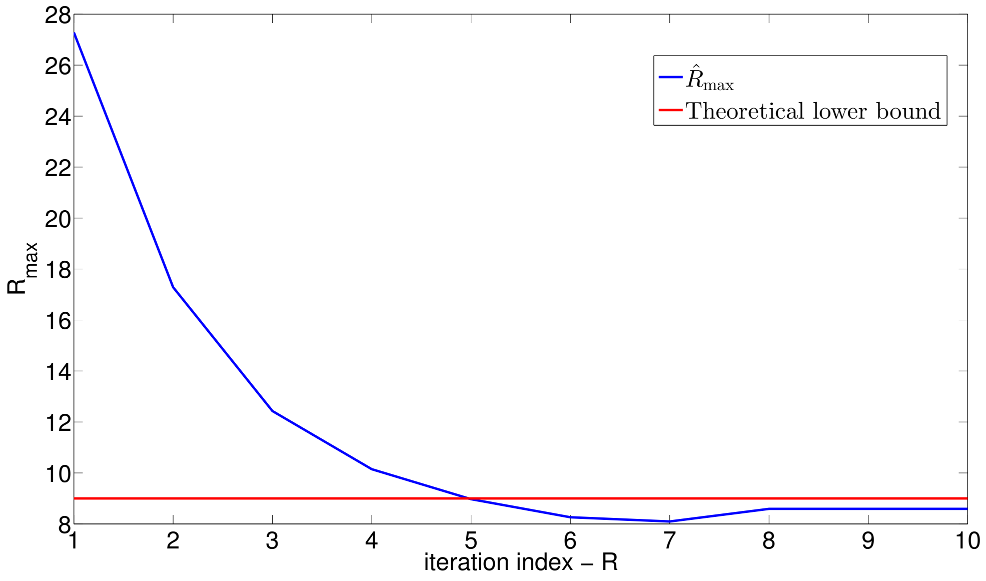


Figure 3.4: Estimates $\hat{R}_{\max}^{(r)}$ [cf. (3.30)] versus iteration index for a dataset with $K = 3$ clusters generated by (3.14) with $D = 1$, cluster means $\{-1, 1, 2\}$, variances $\{0.5, 0.5, 0.5\}$ and number of points per cluster $\{100, 180, 200\}$. The horizontal line represents the $R_{\max}$ value given by Theorem 1 when f is known.

3.4 Numerical Tests

The proposed algorithm is validated using synthetic and some real datasets. SkeVa-SC is compared to the current state-of-the-art algorithm, Scalable sparse subspace clustering (SSSC). The metrics evaluated are:

- Accuracy, i.e. percentage of points correctly clustered:

$$\text{Accuracy} = \frac{\#\text{ of points correctly clustered}}{N}$$

- Normalized mutual information (NMI [65]) between the experimental results and the ground truth labels:

$$\text{NMI} = \frac{I(\Pi; \Pi')}{\max\{H(\Pi), H(\Pi')\}}$$

where $\mathit{\Pi}$ is a random variable that takes values $\{1, 2, \ldots, K\}$ with probabilities $p(\pi_k) = \Pr(\mathit{\Pi} = k) = \frac{N_k}{N}$, i.e. the probabilities of a datapoint belonging to cluster k. Here N_k is the number of datapoints that belong to cluster k, and its value is provided by the algorithms tested. $\mathit{\Pi}'$ is a random variable identical to $\mathit{\Pi}$ but with probabilities derived by the ground-truth labels. $I(\mathit{\Pi}; \mathit{\Pi}')$ denotes the mutual information [61] between the random variables $\mathit{\Pi}$ and $\mathit{\Pi}'$:

$$I(\mathit{\Pi}; \mathit{\Pi}') = \sum_{i,j} p(\pi_i, \pi'_j) \log \frac{p(\pi_i, \pi'_j)}{p(\pi_i)p(\pi'_j)}$$

and $H(\mathit{\Pi})$ denotes the entropy [61] of random variable $\mathit{\Pi}$:

$$H(\mathit{\Pi}) = -\sum_i p(\pi_i) \log p(\pi_i)$$

- Time (in seconds) required for clustering all datapoints.

The bandwidth matrix used for Algorithm 3.3 is $\mathbf{H} = (h(C, n))^2 \mathbf{I}$ with h similar to the one in (A.9), where the unknown quantity pertaining to the curvature of the function to be estimated is replaced with a user-defined constant C, thus:

$$h(C, n) = \left[\frac{CD\phi_{2\mathbf{I}}(\mathbf{0})}{n}\right]^{1/(D+4)} = \left[\frac{CD}{n\,(4\pi)^{D/2}}\right]^{1/(D+4)} \tag{3.31}$$

Furthermore the scoring function ψ used in Algorithm 3.3 is the identity function, $\psi(x) = x$. All results represent the averages of 10 independent Monte Carlo runs.

3.4.1 Synthetic data

Tests on synthetic datasets of dimension $D = 100$ and $D = 500$ with $K = 5$ subspaces are shown in Fig. 3.5 and Fig. 3.6 respectively, where the proposed algorithm is compared to SSSC [54]. The subspaces have dimensions $\{12, 10, 5, 3, 2\}$ and number of datapoints in each subspace is proportional to the subspaces dimension, i.e. $N_k = 200d_k$. The datapoints for each subspace have been generated according to (3.1), where $\{\boldsymbol{\mu}_k = \mathbf{0}\}_{k=1}^K$, the subspace bases $\{\mathbf{U}_k \in \mathbb{R}^{D \times d_k}\}_{k=1}^K$ are randomly generated so that the subspace angle $\eta(i, j)$ between

$\mathcal{S}_i, \mathcal{S}_j$ is at least $\pi/4, \forall(i,j), i = 1, \ldots, K, j = 1, \ldots, K, i \neq j$. The subspace angle $\eta(i,j)$ is defined as:

$$\eta(i,j) = \min_{\boldsymbol{u},\boldsymbol{v}} \left\{ \arccos\left(\frac{|\boldsymbol{u}^T\boldsymbol{v}|}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}\right) | \boldsymbol{u} \in \mathcal{S}_i, \boldsymbol{v} \in \mathcal{S}_j, \right\}$$

The projections $\boldsymbol{y}_i$ of $\boldsymbol{x}_i$'s onto their subspaces are also randomly generated by being drawn from a uniform distribution on the volume of the unit d_k-dimensional hypercube: $\boldsymbol{y}_i \sim \mathcal{U}[-1,1]^{d_k}$, where $\mathcal{U}$ denotes uniform distribution. Finally AWG noise, with variance $\sigma^2 = 0.1$ is added to all datapoints: $\boldsymbol{e}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_D)$. The number of points for the validation stage of Alg.3.3 is $n' = 600$ and the maximum number of iterations is set to $R_{\max} = 100$. As the number of sampled points n increases, so does the clustering accuracy, as well as the normalized mutual information (NMI) between the cluster assignments and the ground truth ones. Both of these metrics for the proposed method are larger than those of SSSC, corroborating the fact that a single random sketch may not be representative of the data to ensure reliable clustering. Additionally, the required time is not much higher than the SSSC algorithm.

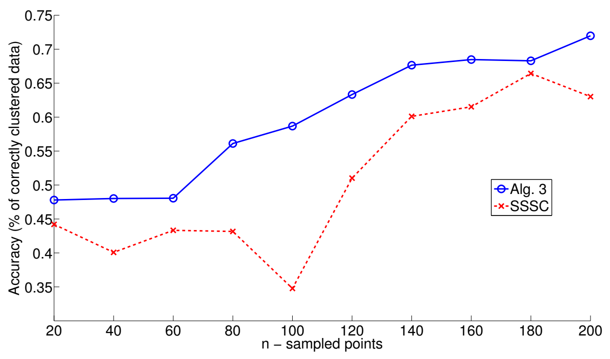
3.4.2 Real data

The real datasets tested are the PenDigits [66] dataset, the Extended Yale Face database [67] and the PokerHand UCI [68] database. The PenDigits dataset includes $N = 10992$ datapoints of dimension $D = 16$, separated into $K = 10$ clusters, with each datapoint representing a handwritten digit. Clusters group same digits, and each cluster contains 250 datapoints.

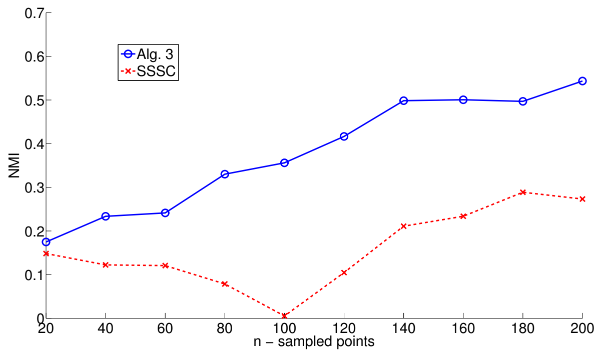
The results for the PenDigits dataset are shown in Fig. 3.7. Here $C = 10^{-3}$ (cf (3.31)), $\mathbf{H} = (h(C,n))^2\mathbf{I}, \mathbf{H}_0 = (\frac{h(C,n)}{2})^2\mathbf{I}, \mathbf{H}' = (h(C,n'))^2\mathbf{I}$, $n' = 700$ and $R_{\max} = 150$. Similar to synthetic results as the number of datapoints increases so does the accuracy and NMI of both Alg. 3.3 and SSSC, with Alg. 3.3 showing higher accuracy and NMI levels at the cost of higher computational time. The accuracy and NMI difference between SSSC and Alg. 3.3 is not as pronounced as in the synthetic datasets, possibly because the PenDigits dataset is uniform: all clusters have the same number of datapoints.

The Extended Yale Face database contains $N = 2414$ face images of $K = 38$ people, each of dimension $D = 2016$. The dimensionality of the data was reduced using PCA by extracting the 114 most important features and Alg. 3.3 and SSSC were tested on the dimensionality reduced and normalized data. Fig. 3.8 shows the results for this dataset. Here $C = 10^{-2}$, $\mathbf{H} = (h(C,n))^2\mathbf{I}, \mathbf{H}_0 = (\frac{h(C,n)}{2})^2\mathbf{I}, \mathbf{H}' = (h(C,n'))^2\mathbf{I}$, $n' = 700$ and $R_{\max} = 150$. Again Alg. 3.3 exhibits higher accuracy and NMI than its one-shot random sampling counterpart with the clustering time being slightly higher.

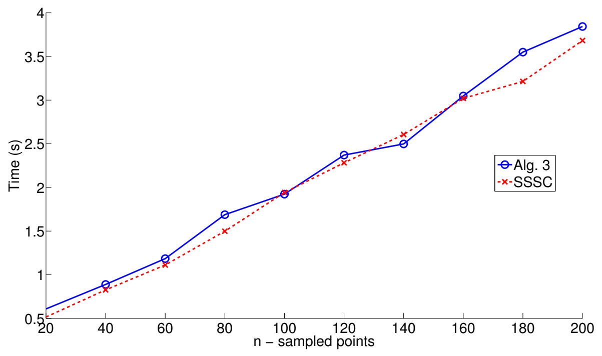
The PokerHand database contains $N = 10^6$ datapoints, belonging to $K = 10$ clusters. Each $D = 10$-dimensional datapoint is a 5-card hand drawn from a deck of 52 cards, with each card being described by its suit(spades, hearts, diamonds and clubs) and rank. Each cluster represents a valid Poker hand. Fig. 3.9 shows the results for this dataset, compared to SSSC. Here $C = 10^{-2}$, $\mathbf{H} = (h(C,n))^2\mathbf{I}, \mathbf{H}_0 = (\frac{h(C,n)}{2})^2\mathbf{I}, \mathbf{H}' = (h(C,n'))^2\mathbf{I}$, $n' = 800$ and $R_{\max} = 150$. Similarly, to the previous datasets Alg. 3.3 exhibits higher accuracy than SSSC, while retaining low clustering time, corroborating the fact that Alg. 3.3 can handle very large datasets. However, NMI is at suprisingly low levels, for both algorithms.



(a) Clustering accuracy

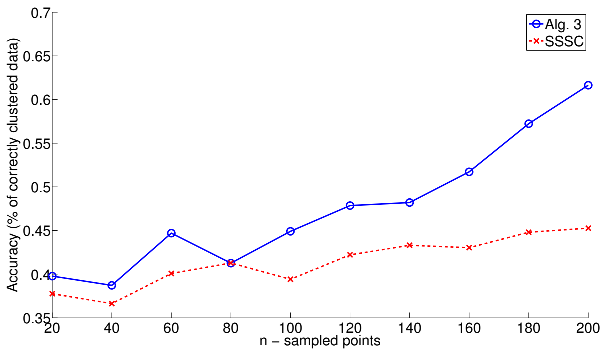


(b) Normalized Mutual Information

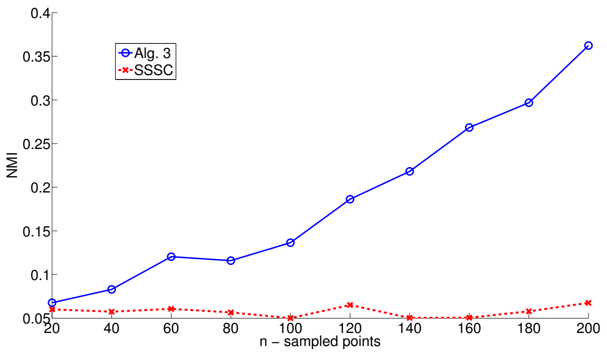


(c) Clustering time (seconds)

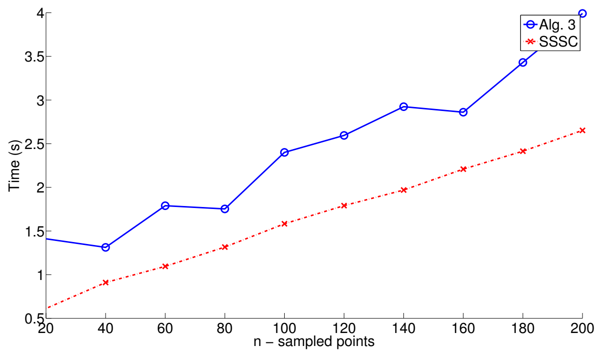
Figure 3.5: Simulated tests on a synthetic dataset with $K = 5$ subspaces, $D = 100$ and $N = 6,400$ datapoints



(a) Clustering accuracy

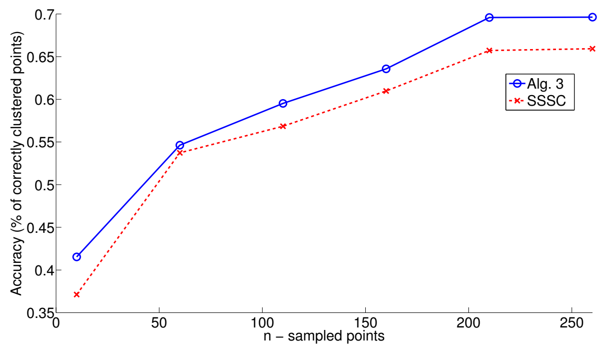


(b) Normalized Mutual Information

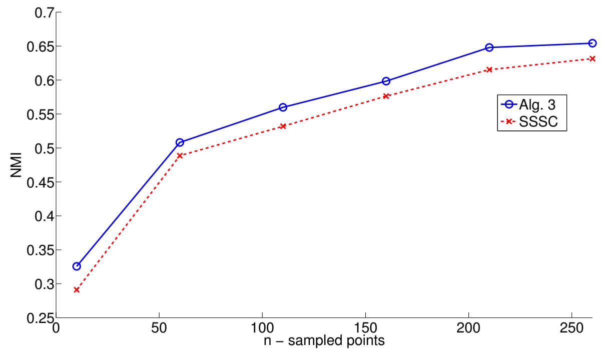


(c) Clustering time (seconds)

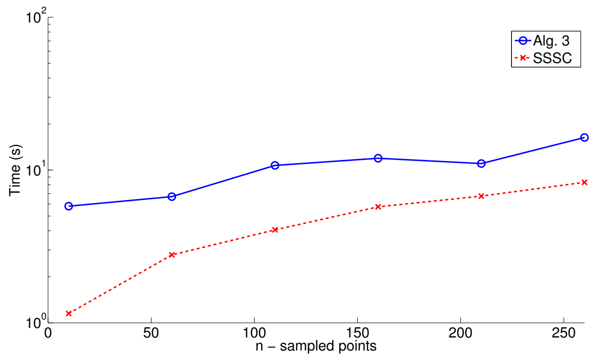
Figure 3.6: Simulated tests on a synthetic dataset with $K = 5$ subspaces, $D = 500$ and $N = 6,400$ datapoints.



(a) Clustering accuracy

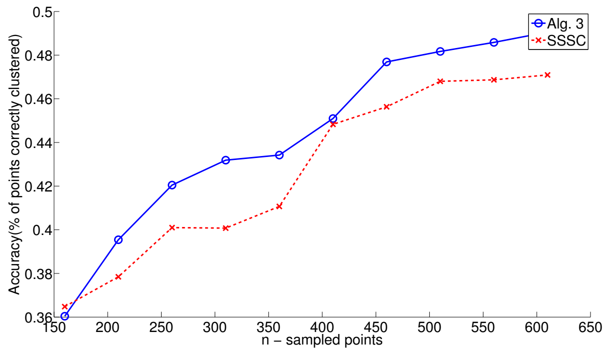


(b) Normalized Mutual Information

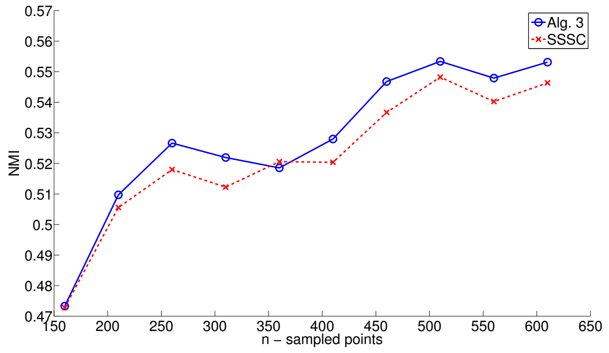


(c) Clustering time (seconds)

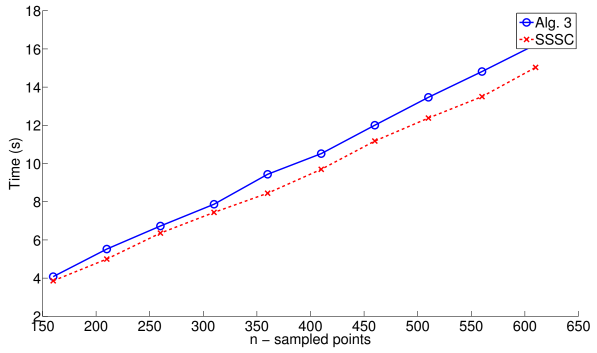
Figure 3.7: Simulated tests on real dataset PenDigits, with $N = 10992$ datapoints dimension $D = 16$ and $K = 10$ clusters.



(a) Clustering accuracy

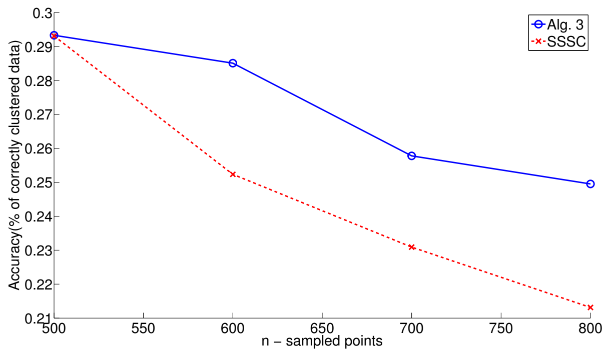


(b) Normalized Mutual Information

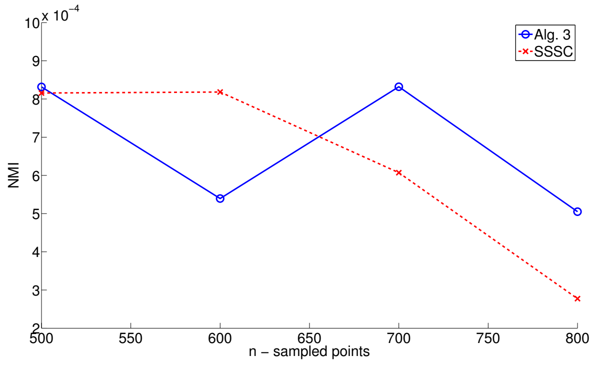


(c) Clustering time (seconds)

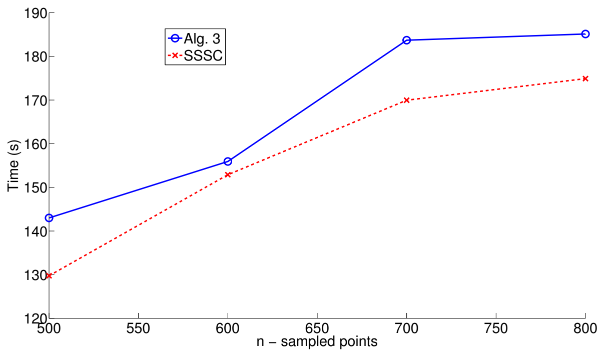
Figure 3.8: Simulated tests on real dataset Extended Yale Face Database B, with $N = 2414$ datapoints dimension $D = 114$ and $K = 38$ clusters.



(a) Clustering accuracy



(b) Normalized Mutual Information



(c) Clustering time (seconds)

Figure 3.9: Simulated tests on real dataset Pokerhand, with $N = 10^6$ datapoints dimension $D = 10$ and $K = 10$ clusters.

Chapter 4

Community Identification in networks using SkeVa

In our era of data deluge, clustering algorithms that do not scale well with the dramatically increasing number of data have to be reconsidered. Spectral clustering, while powerful, is computationally and memory demanding, even for high performance computers. Capitalizing on the relationship between spectral clustering and kernel k-means, the present Chapter introduces a randomized algorithm for identifying communities in large-scale graphs based on a random sketching and validation approach, that enjoys reduced complexity compared to the clairvoyant spectral clustering. Numerical tests on synthetic and real data demonstrate the potential of the proposed approach.

4.1 Preliminaries

4.1.1 The spectral clustering algorithm

Consider an arbitrary undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices of the graph and $\mathcal{E}$ is the set of edges. Denote also the vertices in $\mathcal{G}$ as $\{\nu_i \in \mathcal{V}\}_{i=1}^{N}$, $|\mathcal{V}| = N$, while let $e_{ij} \in \mathcal{E}$ be the edges between vertices ν_i and ν_j, $i \neq j$. Each edge e_{ij} is characterized by a weight w_{ij} that is interpreted as the similarity between vertices ν_i and ν_j. Let $\mathbf{W} \in \mathbb{R}^{N \times N}$ be a matrix (a.k.a. adjacency matrix) such that $[\mathbf{W}]_{ij} = w_{ij}$. The Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$

of $\mathcal{G}$ is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{4.1}$$

where $\mathbf{D} = \operatorname{diag}(d_1, d_2, \ldots, d_N)$, $d_i := \sum_{j=1}^{N} w_{ij}$. Spectral clustering algorithms leverage a key property of the eigenspace of $\mathbf{L}$; when K connected components are present in $\mathcal{G}$, $\mathbf{L}$ possesses K zero eigenvalues, and their corresponding eigenvectors are indicators of these connected components. In general, clusters (or communities in the present context) are not expected to be completely disconnected from the other clusters in the graph. In such a case, clusters are expected to induce eigenvalues close to 0 in $\mathbf{L}$. Spectral clustering algorithms find the K trailing (corresponding to the smallest eigenvalues) eigenvectors $\{\boldsymbol{v}_i \in \mathbb{R}^N\}_{i=1}^{K}$ of $\mathbf{L}$. Then, cluster assignments are obtained by executing K-means on the rows of $\mathbf{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K] \in \mathbb{R}^{N \times K}$. It turns out that this mapping of vertices to rows of $\mathbf{V}$ increases the separability of the clusters and therefore clusters can be obtained by a simple algorithm such as K-means. Variants of the previous algorithm use normalized versions of $\mathbf{L}$ [50, 51]. In the case where the number of clusters K is unknown, one may monitor the behavior of the eigenvalue values and estimate the value of K by identifying the point where a "jump" is observed in eigenvalue magnitudes.

4.1.2 The kernel K-means algorithm

Kernel K-means is a generalization of the popular K-means algorithm tailored for clustering data which are either available in non-vectorial form or they are not linearly separable [6,69]. Instead of dealing with vectors $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^{N}$, kernel K-means works with images of those vectors, $\{\phi(\boldsymbol{x}_i) \in \mathcal{F}\}_{i=1}^{N}$, mapped in a linear space $\mathcal{F}$ of potentially infinite dimension. Kernel K-means minimizes the (non-convex) objective:

$$\sum_{j=1}^{K} \sum_{i=1}^{N} \|\phi(\boldsymbol{x}_i) - \frac{1}{|C_j|} \sum_{l \in C_j} \phi(\boldsymbol{x}_l)\|_{\mathcal{F}}^2 \tag{4.2}$$

where $\{C_i\}_{i=1}^{K}$ denote the clusters, while the centroid of cluster k is denoted by $\mu_k := (1/|C_k|) \sum_{j \in C_k} \phi(\boldsymbol{x}_j)$. Capitalizing on the representer theorem [70], a distance between any datapoint in $\mathcal{F}$ and the centroid of cluster k can be written as

$$\begin{aligned}\|\phi(\boldsymbol{x})_i - \mu_k\|^2 &= \|\phi(\boldsymbol{x}_i) - \frac{1}{|C_k|}\sum_{j\in C_k}\phi(\boldsymbol{x}_j)\|^2 \\ &= [\mathbf{K}]_{ii} - \frac{2}{|C_k|}\sum_{j\in C_k}[\mathbf{K}]_{ij} \\ &\quad + \frac{1}{|C_k|^2}\sum_{j,l\in C_k}[\mathbf{K}]_{jl}\,, \end{aligned} \tag{4.3}$$

where $\mathbf{K} \in \mathbb{R}^{N\times N}$ is a matrix such that $[\mathbf{K}]_{ij} := \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$, with $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ denoting the inner-product of points $\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)$ in $\mathcal{F}$. From (4.3) it is clear that for kernel K-means to operate only knowledge of inner-products (or similarities) between datapoints is required and not the datapoints themselves, effecting thus clustering of non-vectorial or categorical objects (such as vertices of a graph). The optimization problem of minimizing (4.2) is solved in an alternating fashion similar to K-means. The objective (4.2) can be recast in matrix form as follows

$$\begin{aligned}&\min_{\{\mathbf{\Pi},\mathbf{C}\}} \operatorname{tr}(\mathbf{K}) - \operatorname{tr}(\mathbf{C}^{1/2}\mathbf{\Pi}^T\mathbf{K}\mathbf{\Pi}\mathbf{C}^{1/2}) \\ &= \min_{\{\tilde{\mathbf{\Pi}}\}} \operatorname{tr}(\mathbf{K}) - \operatorname{tr}(\tilde{\mathbf{\Pi}}^T\mathbf{K}\tilde{\mathbf{\Pi}}) \\ &= \max_{\tilde{\mathbf{\Pi}}} \operatorname{tr}(\tilde{\mathbf{\Pi}}^T\mathbf{K}\tilde{\mathbf{\Pi}})\,, \end{aligned} \tag{4.4}$$

where $\tilde{\mathbf{\Pi}} := \mathbf{\Pi}\mathbf{C}^{1/2}$; $\mathbf{\Pi} = [\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K] \in \mathbb{R}^{N\times K}$ is the cluster membership matrix such that $\mathbf{\Pi}\mathbf{1} = \mathbf{1}$ ($\mathbf{1}$ is the all-one column vector) and entries of the vectors $\{\boldsymbol{\pi}_k \in \mathbb{R}^N\}_{k=1}^K$ are $[\boldsymbol{\pi}_k]_i = 1$ if point i belongs to cluster k and 0 otherwise (hard clustering); $\mathbf{C} := \operatorname{diag}(1/|C_1|, \ldots, 1/|C_K|) \in \mathbb{R}^{K\times K}$. While this optimization problem is NP-hard due to the binary nature of $\tilde{\mathbf{\Pi}}$, it can be simplified if $\tilde{\mathbf{\Pi}}$ is relaxed to a unitary matrix, i.e., $\tilde{\mathbf{\Pi}}^T\tilde{\mathbf{\Pi}} = \mathbf{I}$. The relaxed problem has a well-known solution [71]

$$\max_{\tilde{\mathbf{\Pi}}} \operatorname{tr}(\tilde{\mathbf{\Pi}}^T\mathbf{K}\tilde{\mathbf{\Pi}}) = \sum_{i=1}^{K}\lambda_i(\mathbf{K}) \tag{4.5}$$

with $\{\lambda_i(\mathbf{K})\}_{i=1}^K$ denoting the K largest eigenvalues of $\mathbf{K}$. The optimal $\tilde{\mathbf{\Pi}}^*$ is then a matrix whose columns are the K eigenvectors corresponding to $\{\lambda_i(\mathbf{K})\}_{i=1}^K$. From (4.5) the connection of kernel K-means and spectral clustering becomes apparent [72], since kernel

K-means is equivalent to maximizing the trace of the similarity matrix $\mathbf{K}$, while spectral clustering minimizes the trace of a quantity involving $\mathbf{I} - \mathbf{K}$. Kernel K-means incurs a complexity of $\mathcal{O}(KN^2I)$, where I is the number of required iterations until convergence.

Remark 1. In both spectral clustering and kernel K-means the choice of similarity measure is critical for the performance of the algorithm.

4.1.3 Prior work

In large social, biological networks, or networks that exhibit community structure, $\mathbf{L}$ is presumed to be sparse. In this case, methods such as Arnoldi/Lanczos iterations [71] can be used to efficiently compute the trailing eigenspace of $\mathbf{L}$, as usually only matrix-vector products are required. Readily available packages that tackle large-scale sparse eigenvalue problems exist [73]. Distributed eigensolvers and parallel versions of k-means [74] can also be used. Care should be taken when the number of communities K is very large. While the trailing eigenvectors of $\mathbf{L}$ can be computed efficiently, the final clustering step would require clustering N K-dimensional vectors, which can prove challenging even for distributed versions of k-means. Multiple approaches that aim to tackle specifically large-scale spectral clustering tasks are available. These approaches come in three major flavors: Parallelization/distributed processing, random sampling and random projections.

A useful overview of performing spectral clustering for sparse Laplacian matrices in parallel is given in [75]. The parallelization can be performed using either MapReduce [34] or MPI [76]. Pre-processing of the data using k-means and random projection trees is investigated in [77]. In this method the preprocessing step reduces the original N datapoints to $M < N$ representatives and performs spectral clustering on these M representatives, which results in a reduced graph which speeds up execution of the spectral clustering algorithm. However, as usually only similiraties between datapoints are given and not the datapoints themselves, algorithms such as kernel k-means or k-medoids, that can work using only similarities have to be employed.

Random sampling and random projections of the data are advocated in [78]. The random projection step involves projecting the data points to a lower dimensional space and computing the similarity matrix from these lower-dimensional representations of the data points. Again, as usually only the similarities are given and not the datapoints themselves, this part of the algorithm cannot provide the needed computational time reduction. Afterwards entries of the similarity matrix are randomly sampled and spectral clustering is performed using this reduced similarity matrix. Nyström's method is proposed in [79] and [80], to form a low-rank similarity matrix, which is enabled by sampling the original similarity matrix and using the similarities between the sampled and non-sampled points. Finding the eigenspace of the new low-rank similarity matrix is much more efficient, however one should perform the sampling carefully, as it can drastically influence the final result. When the similarity matrix is sparse, the Nyström eigenspace can be very similar to the original eigenspace, leading to highly accurate clustering.

Random sampling of the similarity matrix is explored in [81], whereby entries of the similarity matrix are sampled randomly, based on a budget constraint, and all other entries are set to zero. This sparsifies the similarity matrix and leads to faster computation of the eigenvectors. Random sketching is promoted in [82] and [83]. Entries of the similarity matrix are randomly sketched using a random projection matrix to reduce the size of the similarity matrix. This reduction allows for faster computation of the eigenspace.

Large-scale kernel K-means methods can also significantly speed-up the clustering process. Results in [72] have shown that using kernel k-means instead of spectral clustering can reduce the computation time required. Again care should be taken when using kernel K-means. While the Laplacian matrix of a social network might be sparse, the similarity matrix in general is not, possibly increasing the clustering time of kernel K-means (compared to spectral clustering methods), especially in cases where the number of clusters K is small. Methods to scale the kernel K-means algorithm are also available. Random sampling of the kernel matrix is investigated in [16], where the centroids (cluster representatives) are forced to reside on the subspace spanned by those sampled points. Simulated tests demonstrate that the resultant algorithm can tackle large datasets effectively. Parallelization of the kernel k-means algorithm is proposed in [84]. Here, low-dimensional embeddings allow kernel k-means to be used in a distributed fashion using the MapReduce framework.

4.2 The proposed algorithm

The proposed randomized algorithm (tabulated as Algorithm 4.1) capitalizes on the relationship between spectral clustering and kernel K-means and is inspired by random sampling and consensus (RANSAC [25]) techniques. As it utilizes kernel K-means it requires as input the similarity matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$ and the number of communities K. The algorithm works iteratively, for a prescribed number of $R_{\max}$ iterations. At each iteration $n \ll N$ data points are sampled uniformly at random. The corresponding rows and columns of $\mathbf{K}$ are then used to form a smaller matrix $\check{\mathbf{K}} \in \mathbb{R}^{n \times n}$. The algorithm then runs kernel K-means on $\check{\mathbf{K}}$, obtaining a rough sketch of the clusters $\{\check{C}_k\}_{k=1}^{K}$ and their centroids $\{\check{\mu}_k\}_{k=1}^{K}$. This is denoted as the "random sketching" phase of the algorithm. At the next step ("validation phase") the algorithm evaluates how "good" the clustering obtained by the random sketching step is. It does so by sampling another set of $n' \leq N - n$ datapoints and forming the matrix $\check{\mathbf{K}}' \in \mathbb{R}^{n \times (n+n')}$. Notice that $\check{\mathbf{K}}'$ is of the form $\check{\mathbf{K}}' = [\check{\mathbf{K}}|\tilde{\mathbf{K}}]$ where $\tilde{\mathbf{K}} \in \mathbb{R}^{n \times n'}$ contains the similarities between the initially sampled n points and the newly sampled n' points. Using $\check{\mathbf{K}}'$ the newly sampled n' points can be assigned to their closest cluster, and centroids $\{\check{\mu}'_k\}_{k=1}^{K}$ can be then updated accordingly. Afterwards, the points in $\check{\mathbf{K}}$ are checked for their closest cluster according to the updated centroids $\{\check{\mu}'_k\}_{k=1}^{K}$. Points that did not change clusters after the update using n' points are deemed to be in the *validation set* ($\mathcal{VS}$) of the iteration, i.e.,

$$\mathcal{VS}^{(r)} = \left\{\phi(\boldsymbol{x}) \in \check{\mathbf{K}}^{(r)} | \exists k \text{ s.t. } \phi(\boldsymbol{x}) \in \check{C}_k^{(r)} \cap \check{C}_k'^{(r)}\right\} \tag{4.6}$$

where r denotes the iteration index. Each iteration is ranked according to a scoring function f that depends on the validation set. One example of such a scoring function is the cardinality of the validation set: $f(\mathcal{VS}) := |\mathcal{VS}|$.

After $R_{\max}$ iterations the algorithm chooses the one that had the highest score, $r^* := \arg\max f(\mathcal{VS}^{(r)})$. The remaining $N - n$ datapoints are then clustered according to the centroids of $\check{\mathbf{K}}^{(r)}$, $\{\check{\mu}_k^{(r)}\}_{k=1}^{K}$. The proposed algorithm incurs a complexity of $\mathcal{O}(KR_{\max}n^2I + R_{\max}nn')$, which is quadratic only in the number of sampled points n.

Remark 2. Similarly to RANSAC [25], the number of iterations $R_{\max}$ can be related to the reliability of the algorithm. Let p be the probability of drawing a "good" sample at iteration r. Then the probability of getting a "bad" draw is $1-p$. If the total probability of failure for Algorithm 4.1 is q then it is desirable to have a large enough number of iterations R such that

$$(1-p)^R < q\,. \tag{4.7}$$

Hence, the maximum number of iterations can be set according to

$$R_{\max} \geq \frac{\log q}{\log(1-p)}\,. \tag{4.8}$$

Remark 3. As each iteration r is independent from the others, the algorithm admits parallelization through efficient processing schemes such as MPI [76] and MapReduce [34].

Remark 4. In cases where the similarity matrix $\mathbf{K}$ is very sparse, the probability of drawing a good sample is very small. Therefore the algorithm needs to be modified so that sampled entries contain sufficient information (few zero entries) to facilitate clustering.

4.3 Numerical tests

The performance and time complexity of the proposed algorithm are evaluated on synthetic and real datasets. The proposed algorithm is compared to normalized spectral clustering [50], which utilizes Lanczos iterations to compute the trailing eigenvectors of $\mathbf{L}$ in a fast and efficient manner, and kernel K-means on the whole network. The kernel used for Algorithm 4.1 and kernel K-means in all experiments is $\mathbf{K} := (-1/2)\mathbf{JDJ}$ where $\mathbf{D} \in \mathbb{R}^{N\times N}$ is a matrix containing the shortest path distances between all vertex pairs and $\mathbf{J} := \mathbf{I} - (1/N)\mathbf{1}\mathbf{1}^T$ is the double centering operator.

To assess community identification performance, three metrics are used [85]:

- **Average cluster conductance**: Conductance is a measure of how many edges of a cluster point outside it, and it is defined as:

$$g(C_k) = \frac{|\mathcal{U}_{C_k}|}{2|\mathcal{Z}_{C_k}| + |\mathcal{U}_{C_k}|} \tag{4.9}$$

  where $\mathcal{Z}_{C_k}$ is the set of edges in cluster C_k that have both endpoints in C_k, while $\mathcal{U}_{C_k}$ is the set of edges that go out of C_k, i.e., edges that have only one endpoint in C_k.

Algorithm 4.1 Random Sketching and Validation for Spectral Clustering (SkeVa-SC)

```
Input: Gram matrix $\mathbf{K}$, number of clusters $K$, number of points to sample $n$, maximum number of iterations $R_{\max}$
Output: Clustered data
 1: for r = 1 to $R_{\max}$ do
 2:     Sample $n \ll N$ points; call sampled Gram matrix $\check{\mathbf{K}}^{(r)}$.
 3:     Run $K$-means on $\check{\mathbf{K}}^{(r)}$; obtain $K$ clusters $\{C_k^{(r)}\}_{k=1}^K$.
 4:     Sample another set of $n'$ points; form augmented Gram matrix $\check{\mathbf{K}}'^{(r)} \in \mathbb{R}^{n\times(n+n')}$.
 5:     For each of the $n'$ sampled points find the closest cluster and assign them to the corresponding cluster.
 6:     For all sampled points in $\check{\mathbf{K}}^{(r)}$ find their closest cluster.
 7:     $|\mathcal{VS}^{(r)}|$ = number of datapoints that did not change cluster after the augmentation in Step 4.
 8: end for
 9: $r^* = \arg\max f(\mathcal{VS}^{(r)})$.
10: Group the remaining points according to the clusters given by $r^*$.
```

The values of (4.9) are computed for each identified community and are averaged over their number K to produce the average cluster conductance.

- **Modularity**: Modularity is one of the most popular measures used in community identification. It compares the density of edges in subgraphs of the clustered graph with the expected density if there were no community structure, i.e., the graph was random. It is defined as

$$g(C_k) = \frac{1}{4}\left(|\mathcal{Z}_{C_k}| - \mathbb{E}\left[|\mathcal{Z}_{C_k}|\right]\right) \tag{4.10}$$

where the expectation is taken over a random graph with the same degree sequence.

- **Clustering accuracy**: Accuracy is the percentage of correctly classified vertices, given as a number in $[0, 1]$ with 1 corresponding to 100% accuracy. It is used only for synthetic networks as there were no available ground-truth assignments for the real networks used.

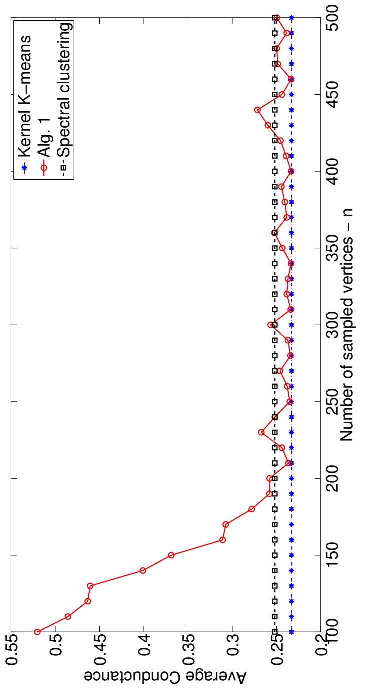
All experiments are conducted via MATLAB [37] on an Intel Core $\mathrm{i}5-4570$ machine clocked at 3.2GHz with 16 Gigabytes of RAM. All reported values are the averages of 10 independent runs.

4.3.1 Synthetic networks

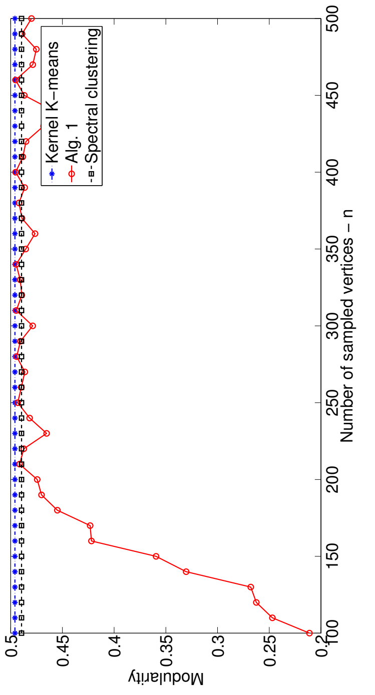
Synthetic networks are generated randomly according to the method described in [86]. This method generates binary adjacency matrix $\mathbf{W} \in \{0,1\}^{N\times N}$ where $[\mathbf{W}]_{ij} = 1$ indicates the existence of an edge between vertices i and j. Figure 4.1 shows the results for a synthetic network with $N = 5,000$ vertices, $K = 9$ communities, and $|\mathcal{E}| = 126,427$ edges, where 300 of the nodes do not belong to a specific community. The average degree of each vertex is 50.04 and the mixing parameter (cf. [86]) is $\mu = 0.3$. This μ defines the number of edges between communities. As a rule of thumb, well-defined communities are created for $\mu < 0.5$. The number of points for validation n' is set to $n' = 700$ and $R_{\max} = 10$. Figure 4.2 shows the results for a synthetic network with $N = 10,000$ vertices, $K = 13$ communities, $|\mathcal{E}| = 250,214$ edges, and 200 of the nodes do not belong to a specific community. The network can be viewed in Fig. 1.5 Again, the average degree of each vertex is 50.04 but the mixing parameter is $\mu = 0.2$. Here the number of points used for validation is set to $n' = 1,000$ and $R_{\max} = 20$. It can be seen that as the number of sampled datapoints n increases, the proposed algorithm approaches the performance of kernel K-means and spectral clustering, while requiring one order of magnitude less computational time.

4.3.2 Real data

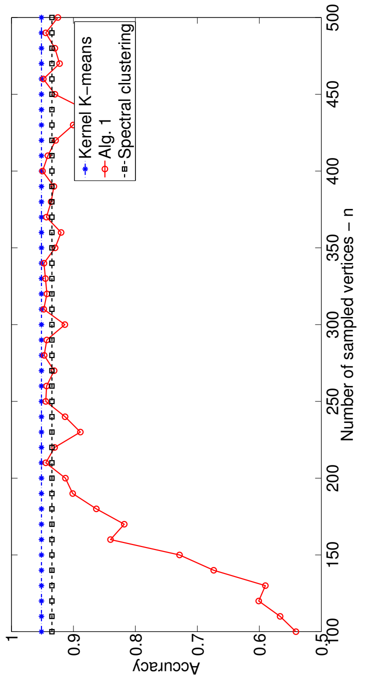
Algorithm 4.1 was also tested on a real datasets. The first dataset consists of the largest connected component of a "Facebook egonet," i.e., the network of friends of a single facebook user [87], pictured in Fig.1.4. Vertices correspond to people and edges denote whether two vertices are "friends" on Facebook. This network contains $N = 744$ vertices, $|\mathcal{E}| = 30,023$ edges, and $K = 5$ communities. Figure 4.3 shows the simulation results for this network. Since the size of the network is small, all algorithms exhibit low clustering time. Clearly, as the number of sampled vertices increases, the conductance and modularity metrics for the proposed approach the levels of kernel K-means and spectral clustering. Here the maximum number of iterations is $R_{\max} = 10$ and $n' = 200$. Figure 4.4 shows results for the largest connected component of an arXiv collaboration network on High energy physics theory [88] with $N = 8,638$ vertices, $|\mathcal{E}| = 24,806$ edges, and $K = 100$ communities. Vertices correspond to authors and edges indicate whether two authors have co-authored a paper. Again, as the number of sampled vertices increases the algorithm exhibits similar behavior with kernel K-means and spectral clustering. Here, $R_{\max} = 10$ and $n' = 800$.



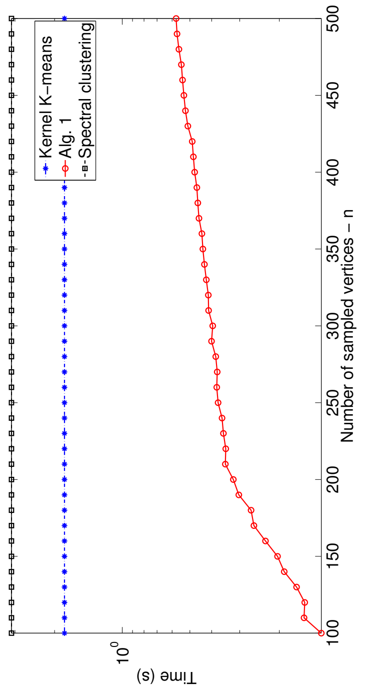
(a) Average conductance



(b) Modularity

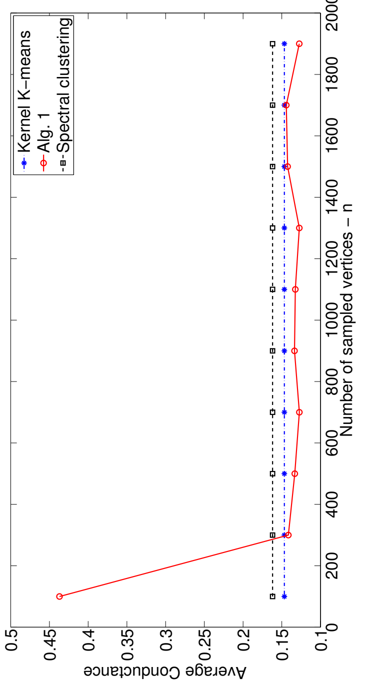


(c) Clustering accuracy

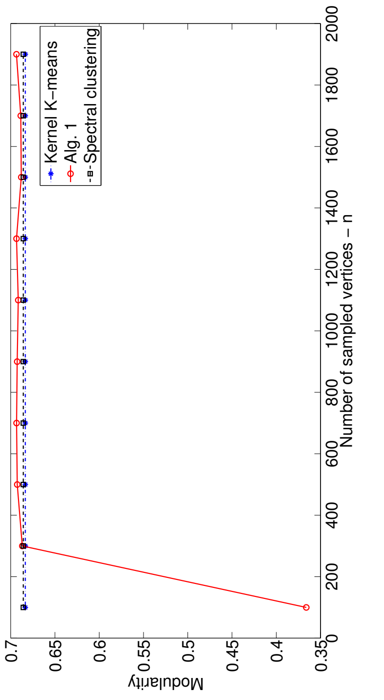


(d) Clustering time (secs.)

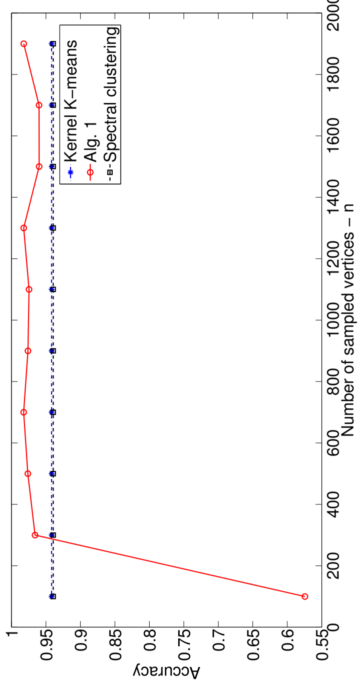
Figure 4.1: Synthetic network with $N = 5,000$ vertices, $|\mathcal{E}| = 126,427$ edges, and $K = 8$ communities.



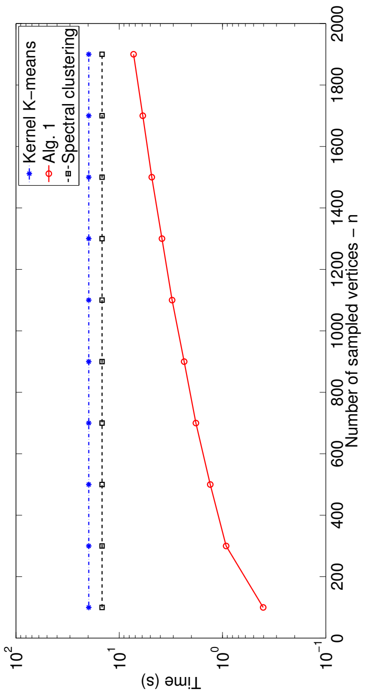
(a) Average conductance



(b) Modularity

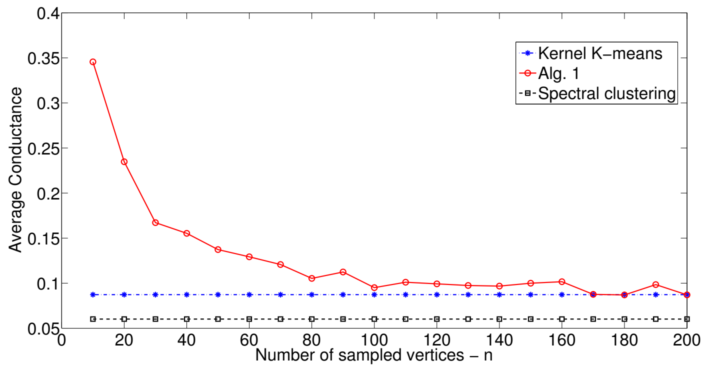


(c) Clustering accuracy

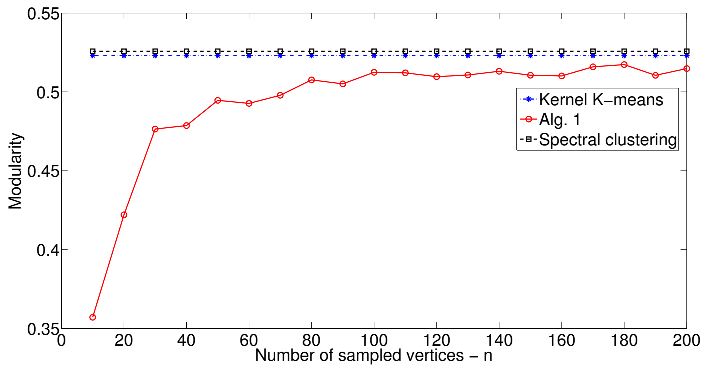


(d) Clustering time (secs.)

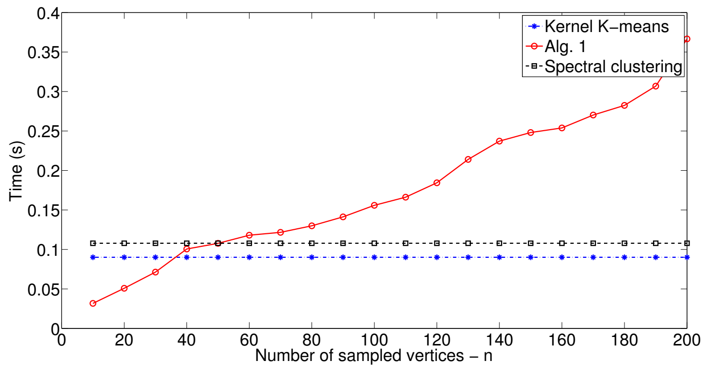
Figure 4.2: Synthetic network with $N = 10,000$ vertices, $|\mathcal{E}| = 250,214$ edges, and $K = 13$ communities.



(a) Average conductance

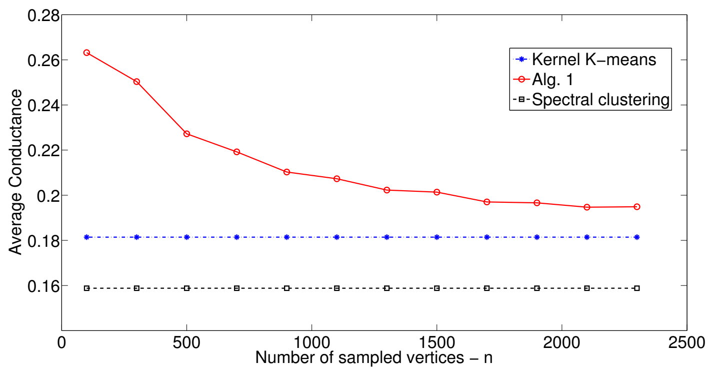


(b) Modularity

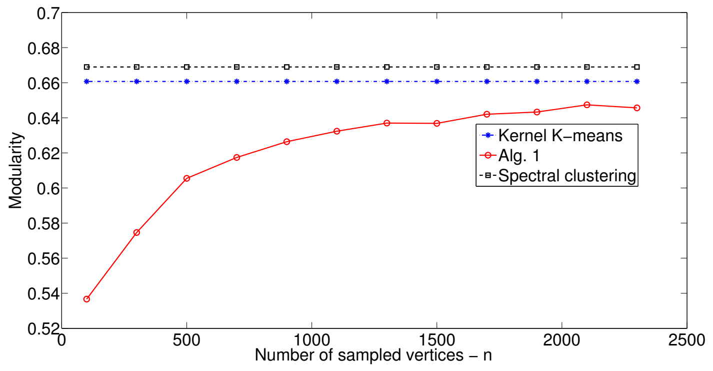


(c) Clustering time (secs.)

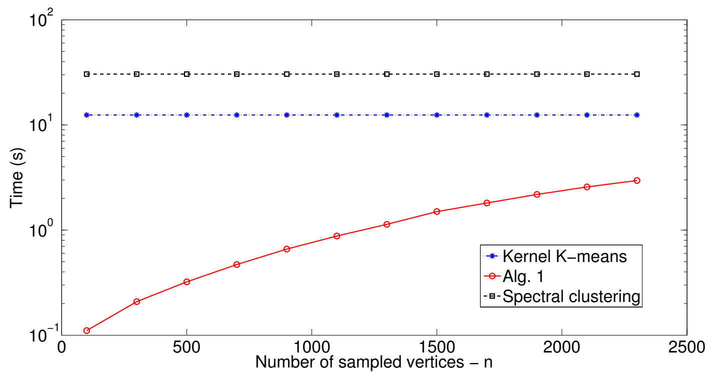
Figure 4.3: Facebook egonet with $N = 744$ vertices, $|\mathcal{E}| = 30,023$ edges, and $K = 5$ communities.



(a) Average conductance



(b) Modularity



(c) Clustering time (secs.)

Figure 4.4: Collaboration network of arXiv High Energy Physics Theory with $N = 8,638$ vertices, $|\mathcal{E}| = 24,806$ edges, and $K = 100$ communities.

Chapter 5

Conclusion and Future Research

Inspired by RANSAC ideas that have well-appreciated merits for outlier-resilient regression, this thesis introduced a novel algorithmic framework for clustering massive numbers of high-dimensional data. Several members of the proposed sketching and validation (SkeVa) family were introduced. The first two members, a batch and a sequential one tailored to streaming modes of operation, required K-means clustering in low-dimensional spaces and/or a small number of data-points. To enable clustering of even nonlinearly separable data, a third member of the family leveraged the kernel trick to cluster linearly separable mapped data. Kernel-based SkeVa also allowed for clustering-based community identification of large networks with a low-computational footprint, by exploiting the relationship between kernel methods and spectral clustering.

A divergence metric was utilized to develop the fourth member of the SkeVa family that bypasses intermediate K-means clustering to trade-off accuracy for reduced complexity, by using ideas from kernel smoothing techniques. This method also allows for subspace clustering of high-volume datasets, by using the Sparse subspace clustering algorithm, but can potentially utilize any other subspace clustering algorithm.

Theoretical bounds on the number of required iterations were provided for this member of the SkeVa family. Extensive numerical tests on synthetic and real data-sets demonstrated the competitive performance of SkeVa algorithms over their state-of-the-art counterparts. Most of the SkeVa algorithms are amenable to parallelization, enabling distributed clustering of high-volume, high-dimensional data and networks.

Future research will focus on rigorous performance analysis of the proposed framework, and on the development of online clustering tools that are able to handle streaming data in an efficient manner, as well as implementations on distributed and parallel platforms. Furthermore, robust variants of SkeVa capable of handling outliers will be pursued. Finally, efficient methods for clustering of overlapping communites will be considered.

Bibliography

[1] K. Cukier, "Data, data everywhere," *The Economist*, 2010. [Online]. Available: http://www.economist.com/node/15557443.

[2] T. Bengtsson, P. Bickel, and B. Li, "Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems," in *Probability and Statistics: Essays in Honor of David A. Freedman.* IMS, 2008, vol. 2, pp. 316–334.

[3] M. Jordan, "On statistics, computation and scalability," *Bernoulli*, vol. 19, no. 4, pp. 1378–1390, 2013.

[4] G. O' Connor. (2014) Moore's law gives way to Bezos' law. [Online]. Available: https://gigaom.com/2014/04/19/moores-law-gives-way-to-bezoss-law/

[5] C. M. Bishop, *Pattern Recognition and Machine Learning.* New York: Springer, 2006.

[6] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel K-means: Spectral clustering and normalized cuts," in *Proc. of SIGKDD Intl. Conf. Knowledge Discovery Data Mining.* ACM, 2004, pp. 551–556.

[7] R. Vidal, "A tutorial on subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2010.

[8] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, pp. 167–256, June 2003.

[9] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. of the Natl. Acad. of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[10] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[11] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[12] H. Zhang and G. Sun, "Feature selection using Tabu search method," *Pattern Recognition*, vol. 35, pp. 701–711, 2002.

[13] B. Yu and B. Yuan, "A more efficient branch and bound algorithm for feature selection," *Pattern Recognition*, vol. 26, no. 6, pp. 883–889, 1993.

[14] C. Chatterjee and V. Roychowdhury, "On self-organizing algorithms and networks for class-separability features," *IEEE Trans. Neural Networks*, vol. 8, no. 3, pp. 663–678, 1997.

[15] V. M. Patel, H. V. Nguyen, and R. Vidal, "Latent space sparse subspace clustering," in *Proc. of ICCV*, Sydney: Australia, 2013, pp. 225–232.

[16] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain, "Approximate kernel K-means: Solution to large-scale kernel clustering," in *Proc. of Knowledge Discovery Data Mining*, San Diego CA: USA, Aug. 2011.

[17] M. Mahoney, "Randomized algorithms for matrices and data," *Found. Trends Machine Learn.*, vol. 3, no. 2, pp. 123–224, 2011.

[18] ——, "Algorithmic and statistical perspectives on large-scale data analysis," in *Combinatorial Scientific Computing*, U. Naumann and O. Schenk, Eds. Chapman and Hall/CRC, 2012, ch. 16, pp. 427–459. [Online]. Available: arXiv:1010.1609v1

[19] K. L. Clarkson and D. P. Woodruff, "Low rank approximation and regression in input sparsity time," in *Proc. of Symposium on Theory of Computing*, June 2013, pp. 81–90. [Online]. Available: arXiv:1207.6365v4

[20] C. Boutsidis, A. Zouzias, M. Mahoney, and P. Drineas, "Randomized dimensionality reduction for K-means clustering," *Computing Research Repository*, 2011. [Online]. Available: arXiv:1110.2897

[21] B. Kang, W. Lim, and K. Jung, "Scalable kernel K-means via centroid approximation," in *Proc. of NIPS*, Granada: Spain, Dec. 2011.

[22] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: A review," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 90–105, 2004.

[23] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

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

[25] M. Fisher and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. ACM*, vol. 24, pp. 381–395, June 1981.

[26] P. H. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[27] D. Nistér, "Pre-emptive RANSAC for live structure and motion estimation," *Machine Vision Appl.*, vol. 16, no. 5, pp. 321–329, 2005.

[28] M. Zuliani, C. S. Kenney, and B. S. Manjunath, "The multiRANSAC algorithm and its application to detect planar homographies," in *Proc. of ICIP*, vol. 3, Genova: Italy, 2005, pp. III–153–6.

[29] R. Toldo and A. Fusiello, "Robust multiple structures estimation with J-linkage," in *Proc. of ECCV*. Marseille: France: Springer, 2008, pp. 537–547.

[30] O. Chum and J. Matas, "Optimal randomized RANSAC," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1472–1482, 2008.

[31] M. P. Wand and M. C. Jones, *Kernel smoothing.* Crc Press, 1994.

[32] C. Micchelli and M. Pontil, "Learning the kernel function via regularization," *J. Machine Learning Research*, vol. 6, pp. 1099–1125, Sept. 2005.

[33] J. C. Principe, *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives.* Springer, 2010.

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

[35] A. Elgohary, A. K. Farahat, M. S. Kamel, and F. Karray, "Embed and conquer: Scalable embeddings for kernel K-means on MapReduce," in *Proc. of Intl. Conf. Data Mining*, Shenzen: China, Dec. 2014.

[36] D. Achlioptas, "Database-friendly random projections," in *Proc. of SIGMOD-SIGACT-SIGART*. ACM, 2001, pp. 274–281.

[37] MATLAB, *version 8.2.0.701 (R2013a).* Natick MA: USA: The MathWorks Inc., 2013.

[38] Minnesota Supercomputing Institute. [Online]. Available: http://www.msi.umn.edu/

[39] Mathworks. Matlab multicore. [Online]. Available: http://www.mathworks.com/discovery/matlab-multicore.html

[40] H.-F. Yu, H.-Y. Lo, H.-P. Hsieh, J.-K. Lou, T. G. Mckenzie, J.-W. Chou, P.-H. Chung, C.-H. Ho, C.-F. Chang, J.-Y. Weng, E.-S. Yan, C.-W. Chang, T.-T. Kuo, P. T. Chang, C. Po, C.-Y. Wang, Y.-H. Huang, Y.-X. Ruan, Y.-S. Lin, S.-D. Lin, H.-T. Lin, and C.-J. Lin, "Feature engineering and classifier ensemble for KDD cup 2010," in *Proc. ACM SIGKDD Conf. Knowledge Discovery Data Mining*, Washington, DC, July 2011.

[41] I. Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge," in *Proc. of Neural Info. Process. Systems*, vol. 4, Whistler BC, Canada, 2004, pp. 545–552.

[42] F. Samaria and A. Harter, "Parameterization of a stochastic model for human face identification," in *Proc. of Workshop on Applications of Computer Vision*, Sarasota FL: USA, Dec. 1994.

[43] AT&T Laboratories Cambridge, UK. The ORL database of faces. [Online]. Available: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[44] R. Zhang and A. I. Rudnicky, "A large scale clustering scheme for kernel K-means," in *Proc. ICPR*, vol. 4, Quebec, Canada, 2002, pp. 289–292.

[45] P. K. Agarwal and N. H. Mustafa, "K-means projective clustering," in *Proc. 23rd ACM SIGMOD-SIGACT-SIGART Symposium.* ACM, 2004, pp. 155–165.

[46] S. Lloyd, "Least squares quantization in pcm," *IEEE Trans. Info. Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[47] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

[48] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy data coding and compression," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1546–1562, 2007.

[49] T. Berger, "Rate-distortion theory," *Encyclopedia of Telecommunications*, 1971.

[50] 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, Dec. 2002.

[51] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[52] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[53] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *Proc. ICML*, 2010, pp. 663–670.

[54] X. Peng, L. Zhang, and Z. Yi, "Scalable sparse subspace clustering," in *Proc. CVPR*, 2013, pp. 430–437.

[55] P. Hall, "Large sample optimality of least squares cross-validation in density estimation," *The Annals of Statistics*, pp. 1156–1174, 1983.

[56] D. W. Scott and G. R. Terrell, "Biased and unbiased cross-validation in density estimation," *J. American Statistical Association*, vol. 82, no. 400, pp. 1131–1146, 1987.

[57] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *J. Royal Statistical Society (Series B)*, pp. 683–690, 1991.

[58] P. Hall, J. S. Marron, and B. U. Park, "Smoothed cross-validation," *Probability Theory and Related Fields*, vol. 92, no. 1, pp. 1–20, 1992.

[59] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[60] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.* MIT Press, 2006.

[61] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* John Wiley & Sons, 2012.

[62] F. Itakura and S. Saito, "Analysis synthesis telephony based on the maximum likelihood method," in *Proc. Intern. Congress Acoustics*, vol. 17, 1968, pp. C17–C20.

[63] S. Acharyya, A. Banerjee, and D. Boley, "Bregman divergences and triangle inequality." in *Proc. of SDM.* SIAM, 2013, pp. 476–484.

[64] S. I. Amari, O. E. Barndorff-Nielsen, R. E. Kass, S. L. Lauritzen, and C. R. Rao, "Differential geometry in statistical inference," *Lecture Notes-Monograph Series*, pp. i–240, 1987.

[65] D. Cai, X. He, and J. Han, "Document clustering using locality preserving indexing," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 12, pp. 1624–1637, 2005.

[66] F. Alimoglu and E. Alpaydin, "Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition," in *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96*. Citeseer, 1996.

[67] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 6, pp. 643–660, 2001.

[68] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[69] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning.* New York: Springer, 2001.

[70] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Computational learning theory.* Springer, 2001, pp. 416–426.

[71] G. H. Golub and C. F. V. Loan, *Matrix Computations.* JHU Press, 2012, vol. 3.

[72] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors: A multilevel approach," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[73] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM, 1998, vol. 6.

[74] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on MapReduce," in *Proc. of the 1st Intl. Conf. on Cloud Computing*, Beijing, China, Dec. 2009.

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

[76] M. Snir, *MPI–the Complete Reference: The MPI Core.* MIT press, 1998, vol. 1.

[77] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *Proc. of the 15th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining.* San Jose, CA: ACM, Aug 2009, pp. 907–916.

[78] T. Sakai and A. Imiya, "Fast spectral clustering with random projection and sampling," in *Proc. of the 6th Intl. Conf. on Machine Learning and Data Mining in Pattern Recognition*, Leipzig, Germany, Jul. 2009.

[79] L. Wang, C. Leckie, K. Ramamohanarao, and J. Bezdek, "Approximate spectral clustering," in *Proc. of the 13th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, Bangkok, Thailand, Apr. 2009.

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

[81] O. Shamir and N. Tishby, "Spectral clustering on a budget," in *Intl. Conf. on Artificial Intelligence and Statistics*, San Diego,CA, May 2011, pp. 661–669.

[82] A. Gittens, P. Kambadur, and C. Boutsidis, "Approximate spectral clustering via randomized sketching," *arXiv preprint arXiv:1311.2854*, 2013.

[83] F. Lin and W. W. Cohen, "Power iteration clustering," in *Proc. of the 27th Intl. Conf. on Machine Learning*, Haifa, Israel, Jun. 2010.

[84] A. Elgohary, A. K. Farahat, M. S. Kamel, and F. Karray, "Embed and conquer: Scalable embeddings for kernel k-means on MapReduce," in *SIAM Intl. Conf. on Data Mining*, Philadelphia, PA, Apr. 2014.

[85] J. J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[86] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, no. 4, p. 046110, 2008.

[87] J. Leskovec, "Social circles: Facebook," *Stanford Network Analysis Project*, 2012. [Online]. Available: http://snap.stanford.edu/data/egonets-Facebook.html

[88] ——, "High energy physics - phenomenology collaboration network," *Stanford Network Analysis Project*, 2012. [Online]. Available: http://snap.stanford.edu/data/ca-HepPh.html

[89] L. Devroye, "Exponential inequalities in nonparametric estimation," in *Nonparametric Functional Estimation and Related Topics*. Springer, 1991, pp. 31–44.

Appendices

Appendix A

Kernel Smoothing

Kernel density estimation or kernel smoothing [31] is a nonparametric method used to estimate an unknown pdf f from a finite number n of its samples. Kernel density estimators, similarly to the histogram and unlike parametric estimators, make no assumptions about the unknown pdf. Due to the use of kernel functions, instead of rectangular bins, kernel smoothers enjoy faster convergence rates than the histogram estimator as n tends to infinity [31]. However, the convergence rate is smaller than that for parametric estimators. For datapoints $\{\boldsymbol{x}_i\}_{i=1}^n \sim f$, sampled from a multivariate pdf f, the kernel density estimator of f is given by [31] :

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{x}_i) \tag{A.1}$$

where $K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{x}_i)$ denotes a pre-defined kernel function centered at $\boldsymbol{x}_i$, with bandwidth matrix $\mathbb{R}^{D\times D} \ni \mathbf{H} \succ \mathbf{0}$. This bandwidth matrix allows for different amounts of smoothing across different dimensions. Typically, $K_{\mathbf{H}}(\boldsymbol{x})$ is chosen to be a density so that $\hat{f}(\boldsymbol{x})$ is also a density. In this case, the role of bandwidth can be understood more clearly as $\mathbf{H}$ takes the role of the covariance matrix of the density $K_{\mathbf{H}}(\boldsymbol{x})$.

The performance of kernel smoothers is usually assessed using the mean integrated square error (MISE):

$$\text{MISE}(\hat{f}; \mathbf{H}) := \mathbb{E}\left[\int \left(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})\right)^2 d\boldsymbol{x}\right]. \tag{A.2}$$

Here $\int$ denotes integration across the D-dimensional Euclidean space, and $d\boldsymbol{x}$ stands for $dx_1 dx_2 \ldots dx_D$, where $\boldsymbol{x} := [x_1, x_2, \ldots, x_D]^\top$. The choice of bandwidth matrix $\mathbf{H}$ has a notable effect on MISE. As a result, it makes sense to find values of $\mathbf{H}$ that allow MISE to stay in relatively low levels. Note that

$$\mathbb{E}[\hat{f}(\boldsymbol{x})] = \int K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{y}) f(\boldsymbol{y}) d\boldsymbol{y} = \int K_{\mathbf{H}}(\boldsymbol{u}) f(\boldsymbol{x} - \mathbf{H}^{1/2}\boldsymbol{u}) d\boldsymbol{u} \tag{A.3}$$

Letting $n \to \infty$ and $\mathbf{H} := h^2\mathbf{I}_D$, where $h > 0$ and $\mathbf{I}_D$ is the $D \times D$ identity matrix, it is possible to minimize (A.2) with respect to h. To this end h has to be defined as a function of n satisfying:

$$\lim_{n\to\infty} h = 0, \quad \lim_{n\to\infty} nh = \infty \tag{A.4}$$

Then the Taylor expansion of f around a point $\boldsymbol{x}$ is

$$f(\boldsymbol{x} - h\boldsymbol{u}) = f(\boldsymbol{x}) - h\boldsymbol{u}^\top \mathcal{D}_f(\boldsymbol{x}) + \frac{1}{2}h^2\boldsymbol{u}^\top \mathcal{H}_f(\boldsymbol{x})\boldsymbol{u} + o(h^2\boldsymbol{u}^\top\boldsymbol{u}) \tag{A.5}$$

where $\mathcal{D}_f(\boldsymbol{x}) = [\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \frac{\partial f(\boldsymbol{x})}{\partial x_2}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_D}]$ is a vector of first-order partial derivatives of f and $\mathcal{H}_f(\boldsymbol{x}) \in \mathbb{R}^{D\times D}$ is the Hessian matrix of f whose entry (i,j) is $[\mathcal{H}_f(\boldsymbol{x})]_{ij} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j}$. Using (A.2),(A.3),(A.4) and (A.5) and solving for h the (asymptotically optimal) value is given by [31]:

$$h^* = \left[\frac{D\int K^2(\boldsymbol{x})d\boldsymbol{x}}{n\int \boldsymbol{x}\boldsymbol{x}^\top K(\boldsymbol{x})d\boldsymbol{x}\int(\nabla^2 f(\boldsymbol{x}))^2 d\boldsymbol{x}}\right]^{1/(D+4)} . \tag{A.6}$$

where $\int \boldsymbol{x}\boldsymbol{x}^\top K(\boldsymbol{x} = \left(\int x_i^2 K(\boldsymbol{x})d\boldsymbol{x}\right)\mathbf{I}_D$ and $\nabla^2 f(\boldsymbol{x}) = \sum_{i=1}^D \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i^2}$. In addition, when the density f to be estimated is a mixture of Gaussians, and the kernel is chosen to be a Gaussian density $K_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{x}_i) := \phi_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{x}_i)$, where

$$\phi_{\mathbf{H}}(\boldsymbol{x} - \boldsymbol{x}_i) := \frac{1}{(2\pi)^{D/2}|\mathbf{H}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_i)^\top \mathbf{H}^{-1}(\boldsymbol{x} - \boldsymbol{x}_i)\right) , \tag{A.7}$$

with mean $\boldsymbol{x}_i$ and covariance matrix $\mathbf{H}$, it is possible to compute (A.2) in closed form. This closed-form computation is enabled by the following property of Gaussian densities [33]:

$$\int \phi_{\mathbf{H}_i}(\boldsymbol{x} - \boldsymbol{x}_i)\phi_{\mathbf{H}_j}(\boldsymbol{x} - \boldsymbol{x}_j)d\boldsymbol{x} = \phi_{(\mathbf{H}_i + \mathbf{H}_j)}(\boldsymbol{x}_i - \boldsymbol{x}_j) , \tag{A.8}$$

that is, the convolution of two Gaussian densities is also a Gaussian density. In the case where $K_{\mathbf{H}}(\boldsymbol{x}) = \phi_{\mathbf{I}_D}(\boldsymbol{x})$, the optimal asymptotic bandwidth of (A.6) becomes

$$h^* = \left[\frac{D\phi_{2\mathbf{I}_D}(\mathbf{0})}{n\int(\nabla^2 f(\boldsymbol{x}))^2 d\boldsymbol{x}}\right]^{1/(D+4)} . \tag{A.9}$$

Appendix B

Soft Kernel K-means

Since this appendix deals with $N \gg$, the clustering schemes of Sec. 2.1 will be applied here to a reduced number $\check{\nu}$ of $D\times 1$ vectors. Let $\check{\boldsymbol{X}}^{(r)} \in \mathbb{R}^{D\times\check{\nu}}$ denote the subset of data obtained by sketching columns of $\boldsymbol{X}$. In this context, (2.2) proceeds as follows. For $i = 1, 2, \ldots,$

[i-a] **Update data-cluster associations:** For $n = 1, \ldots, \check{\nu}$,

$$\boldsymbol{\pi}_n[i] \in \arg \min_{\substack{\boldsymbol{\pi}\in[0,1]^K \\ \mathbf{1}^\top\boldsymbol{\pi}=1}} \delta\left(\varphi(\boldsymbol{x}_n^{(r)}), \sum\nolimits_k \pi_k \boldsymbol{c}_k[i]\right) + \rho(\boldsymbol{\pi}) . \tag{B.1a}$$

[i-b] **Update cluster centroids:**

$$\{\boldsymbol{c}_k[i+1]\}_{k=1}^K \in \arg \min_{\{\boldsymbol{c}_k\}\subset\mathcal{H}} \sum_{n=1}^{\check{\nu}} \delta\left(\varphi(\boldsymbol{x}_n^{(r)}), \sum_k \big[\boldsymbol{\pi}_n[i]\big]_k \boldsymbol{c}_k\right) . \tag{B.1b}$$

Recall that given a kernel κ, if $\varphi(\boldsymbol{x}) := \kappa(\boldsymbol{x}, \cdot)$, then inner products in $\mathcal{H}$ can be obtained as kernel evaluations: $\langle\varphi(\boldsymbol{x}) \mid \varphi(\boldsymbol{x}')\rangle_{\mathcal{H}} = \langle\kappa(\boldsymbol{x}, \cdot) \mid \kappa(\boldsymbol{x}', \cdot)\rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}, \boldsymbol{x}')$, where $\langle\cdot \mid \cdot\rangle_{\mathcal{H}}$ denotes the inner product in $\mathcal{H}$. Moreover, function δ is chosen as

$$\delta\left(\varphi(\boldsymbol{x}_n^{(r)}), \sum\nolimits_k [\boldsymbol{\pi}_n]_k \boldsymbol{c}_k\right) = \left\|\varphi(\boldsymbol{x}_n^{(r)}) - \sum\nolimits_k [\boldsymbol{\pi}_n]_k c_k\right\|_{\mathcal{H}}^2 ,$$

with $\|\cdot\|_{\mathcal{H}} := \langle\cdot \mid \cdot\rangle_{\mathcal{H}}^{1/2}$. It can be shown then by the Representer's theorem [5] that due to the limited number of data $\{\varphi(\boldsymbol{x}_n^{(r)})\}_{n=1}^{\check{\nu}'}$, looking for a solution of (B.1b) in $\mathcal{H}$ is equivalent to looking for one in the low-dimensional linear subspace $\mathcal{H}^{(r)} := \text{span}\{\varphi(\boldsymbol{x}_1^{(r)}), \ldots, \varphi(\boldsymbol{x}_{\check{\nu}}^{(r)})\}$, of rank $\leq \check{\nu}$, and where span stands for the linear span of a set of vectors. For notational convenience, let $\Phi(\check{\boldsymbol{X}}^{(r)}) := [\varphi(\boldsymbol{x}_1^{(r)}), \ldots, \varphi(\boldsymbol{x}_{\check{\nu}}^{(r)})]$, and $\Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{b} := \sum_{n=1}^{\check{\nu}} b_n \varphi(\boldsymbol{x}_n^{(r)})$, for any $\boldsymbol{b} \in \mathbb{R}^{\check{\nu}}$.

Centroid $\boldsymbol{c}_k$ belongs to $\mathcal{H}^{(r)}$, and can be expressed as a linear superposition of $\Phi(\check{\boldsymbol{X}}^{(r)})$. Specifically, there exists $\boldsymbol{b}_k \in \mathbb{R}^{\check{\nu}}$ s.t. $\boldsymbol{c}_k = \Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{b}_k$. Upon defining $\boldsymbol{B} := [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_K]$, then $[\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K] = \Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{B}$. Moreover,

$$\delta\left(\varphi(\boldsymbol{x}_n^{(r)}), \sum\nolimits_k [\boldsymbol{\pi}_n]_k c_k\right) = \left\|\varphi(\boldsymbol{x}_n^{(r)}) - \Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{B}\boldsymbol{\pi}_n\right\|_{\mathcal{H}}^2$$

which can be also obtained through kernel evaluations. Letting $\boldsymbol{K}^{(r)}$ denote the $\check{\nu}\times\check{\nu}$ kernel matrix with (n,n')th entry $[\boldsymbol{K}^{(r)}]_{nn'} := \kappa(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}_{n'}^{(r)})$, and $\boldsymbol{k}_{n'^{(r)}}^{(r)}$ the $\check{\nu}\times 1$ vector with nth entry $[\boldsymbol{k}_{n'^{(r)}}^{(r)}]_n := \kappa(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}_{n'}^{(r)})$, it follows from the linearity of inner products that for any $\check{\nu}\times 1$ vector $\boldsymbol{\xi}$, $\langle\Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{\xi} \mid \varphi(\boldsymbol{x}_n^{(r)})\rangle_{\mathcal{H}} = \boldsymbol{\xi}^\top\boldsymbol{k}_{n^{(r)}}^{(r)}$ and $\langle\Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{\xi} \mid \Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{\xi}\rangle_{\mathcal{H}} = \boldsymbol{\xi}^\top\boldsymbol{K}^{(r)}\boldsymbol{\xi}$. As such, the quadratic term in (B.2) becomes

$$\begin{aligned}
&\left\|\varphi(\boldsymbol{x}_n^{(r)}) - \Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{B}\boldsymbol{\pi}_n\right\|_{\mathcal{H}}^2 \\
&= \left\langle\varphi(\boldsymbol{x}_n^{(r)}) \,\middle|\, \varphi(\boldsymbol{x}_n^{(r)})\right\rangle_{\mathcal{H}} - 2\left\langle\Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{B}\boldsymbol{\pi}_n \,\middle|\, \varphi(\boldsymbol{x}_n^{(r)})\right\rangle_{\mathcal{H}} \\
&\quad + \left\langle\Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{B}\boldsymbol{\pi}_n \,\middle|\, \Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{B}\boldsymbol{\pi}_n\right\rangle_{\mathcal{H}} \\
&= \kappa(\boldsymbol{x}_n^{(r)}, \boldsymbol{x}_n^{(r)}) - 2\boldsymbol{\pi}_n^\top\boldsymbol{B}^\top\boldsymbol{k}_{n^{(r)}}^{(r)} + \boldsymbol{\pi}_n^\top\boldsymbol{B}^\top\boldsymbol{K}^{(r)}\boldsymbol{B}\boldsymbol{\pi}_n
\end{aligned} \tag{B.2}$$

which shows that kernel K-means in (B.1) boils down to solving a finite-dimensional optimization task w.r.t. $\boldsymbol{B}$ and $\boldsymbol{\Pi} := [\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{\check{\nu}}]$.

Moreover, distances in $\mathcal{H}$ between $\varphi(\boldsymbol{x}_n^{(r')}) \in \Phi(\check{\boldsymbol{X}}^{(r')})$ and centroids $\check{c}_k^{(r)} = \Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{b}_k^{(r)}$ needed in step 5 of Alg. 2.3 can be efficiently computed because they are also expressible in terms of kernel evaluations as follows;

$$\begin{aligned}
&\left\|\varphi(\boldsymbol{x}_n^{(r')}) - \Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{b}_k^{(r)}\right\|_{\mathcal{H}}^2 \\
&= \left\langle\varphi(\boldsymbol{x}_n^{(r')}) \,\middle|\, \varphi(\boldsymbol{x}_n^{(r')})\right\rangle_{\mathcal{H}} - 2\left\langle\Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{b}_k^{(r)} \,\middle|\, \varphi(\boldsymbol{x}_n^{(r')})\right\rangle_{\mathcal{H}} \\
&\quad + \left\langle\Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{b}_k^{(r)} \,\middle|\, \Phi(\check{\boldsymbol{X}}^{(r)})\boldsymbol{b}_k^{(r)}\right\rangle_{\mathcal{H}} \\
&= \kappa(\boldsymbol{x}_n^{(r')}, \boldsymbol{x}_n^{(r')}) - 2\boldsymbol{b}_k^{(r)\top}\boldsymbol{k}_{n^{(r')}}^{(r)} + \boldsymbol{b}_k^{(r)\top}\boldsymbol{K}^{(r)}\boldsymbol{b}_k^{(r)}\,.
\end{aligned}$$

Appendix C

Performance bounds using the integrated squared error

Upon definition of d, $\forall f, g \in \mathcal{X}$,

$$d(f,g) := \sqrt{d_{\text{ISE}}(f,g)} := \sqrt{\int (f(\boldsymbol{x}) - g(\boldsymbol{x}))^2 \, d\boldsymbol{x}}\,. \tag{C.1}$$

using (C.1), the concavity of $\sqrt{\cdot}$, and Jensen's inequality [61], an upper bound on $\mathbb{E}[d(\cdot,\cdot)]$ is provided as follows:

$$\begin{aligned}\mathbb{E}[d(\cdot,\cdot)] &= \mathbb{E}\left[\sqrt{d_{\text{ISE}}(\cdot,\cdot)}\right] \leq \sqrt{\mathbb{E}\left[d_{\text{ISE}}(\cdot,\cdot)\right]} \\ &= \sqrt{\mathbb{E}\left[d^2(\cdot,\cdot)\right]}\,,\end{aligned} \tag{C.2}$$

and consequently, $\mathbb{E}^2[d(\cdot,\cdot)] \leq \mathbb{E}[d^2(\cdot,\cdot)]$.

Based on (3.25), define $\delta_0 := d(f_0, \hat{f})$, which is a random variable given its dependance on $\hat{f}$. By the triangle inequality property, δ_0 can be bounded as

$$\delta_0 \leq d(f, \hat{f}) + \delta'\,. \tag{C.3}$$

While $d(f, \hat{f})$ is a quantity that depends on $\hat{f}$, and it is thus random, it will be further bounded by deterministic quantities, yielding a deterministic bound on δ_0.

Distance $\delta' := d(f, f_0)$ can be expressed in closed form using As. 1 and property (A.8):

$$\begin{aligned}
\delta'^2 = d^2(f, f_0) &= \int \left(f(\boldsymbol{x}) - f_0(\boldsymbol{x})\right)^2 d\boldsymbol{x} \\
&= \int f^2(\boldsymbol{x})d\boldsymbol{x} + \int f_0^2(\boldsymbol{x})d\boldsymbol{x} - 2\int f(\boldsymbol{x})f_0(\boldsymbol{x})d\boldsymbol{x} \\
&= \int \sum_{\ell}\sum_{\ell'} w_\ell w_{\ell'} \phi_{\boldsymbol{\Sigma}_\ell}(\boldsymbol{\mu}_\ell - \boldsymbol{x})\phi_{\boldsymbol{\Sigma}_{\ell'}}(\boldsymbol{\mu}_{\ell'} - \boldsymbol{x}) \\
&\quad + \int \phi_{\mathbf{H}_0}(\boldsymbol{\mu}_0 - \boldsymbol{x})\phi_{\mathbf{H}_0}(\boldsymbol{\mu}_0 - \boldsymbol{x}) \\
&\quad - 2\int \sum_{\ell} w_l \phi_{\boldsymbol{\Sigma}_\ell}(\boldsymbol{\mu}_\ell - \boldsymbol{x})\phi_{\mathbf{H}_0}(\boldsymbol{\mu}_0 - \boldsymbol{x}) \\
&= \mathbf{w}^\top \boldsymbol{\Omega}_0 \mathbf{w} + \frac{1}{(4\pi)^{D/2}|\mathbf{H}_0|^{1/2}} \\
&\quad - 2\sum_{\ell} w_\ell \phi_{\boldsymbol{\Sigma}_\ell + \mathbf{H}_0}(\boldsymbol{\mu}_\ell - \boldsymbol{\mu}_\mathbf{0}),
\end{aligned} \tag{C.4}$$

where $\mathbf{w} := [w_1, w_2, \ldots, w_L]^\top$ is formed by the mixing coefficients of (3.14), and $|\mathbf{H}|$ denotes the determinant of $\mathbf{H}$. Distances $d(\hat{f}, f)$ and $d(\hat{f}, f_0)$ are random variables since they depend on $\hat{f}$, which in turn depends on the randomly drawn data $\boldsymbol{x}_i$. As such, only their expectation, w.r.t. the true data pdf f can be computed in closed form. As data are drawn independently per iteration, the expectations do not depend on the iteration r. Under As. 1, $\mathbb{E}[d^2(\hat{f}, f_0)]$ and $\mathbb{E}[d^2(f, \hat{f})]$ can be also expressed in closed form [31] as

$$\begin{aligned}
\mathbb{E}[d^2(\hat{f}, f_0)] = \mathbb{E}[d_{\mathrm{ISE}}(\hat{f}, f_0)] &= \mathbb{E}\left[\int \left(\hat{f}(\boldsymbol{x}) - f_0(\boldsymbol{x})\right)^2 d\boldsymbol{x})\right] \\
&= \int f_0^2(\boldsymbol{x})d\boldsymbol{x} + \mathbb{E}\left[\int \hat{f}^2(\boldsymbol{x})d\boldsymbol{x}\right] \\
&\quad - 2\,\mathbb{E}\left[\int f_0(\boldsymbol{x})\hat{f}(\boldsymbol{x})d\boldsymbol{x}\right] \\
&= \frac{1}{(4\pi)^{D/2}|\mathbf{H}_0|^{1/2}} + \int \mathbb{E}\left[\hat{f}^2(\boldsymbol{x})d\boldsymbol{x}\right] \\
&\quad - 2\,\mathbb{E}\left[\int f_0(\boldsymbol{x})\hat{f}(\boldsymbol{x})d\boldsymbol{x}\right] \\
&= \frac{1}{(4\pi)^{D/2}|\mathbf{H}_0|^{1/2}} + \frac{1}{n}\frac{1}{(4\pi)^{D/2}|\mathbf{H}|^{1/2}} \\
&\quad + \left(1 - \frac{1}{n}\right)\mathbf{w}^\top \boldsymbol{\Omega}_2 \mathbf{w} \\
&\quad - 2\sum_{\ell} w_\ell \phi_{\mathbf{H}+\mathbf{H}_0+\boldsymbol{\Sigma}_\ell}(\boldsymbol{\mu}_\ell - \boldsymbol{\mu}_0).
\end{aligned} \tag{C.5}$$

and

$$\begin{aligned}\mathbb{E}[d^2(\hat{f},f)] &= \mathbb{E}[d_{\mathrm{ISE}}(\hat{f},f)] \\ &= \frac{1}{n(4\pi)^{D/2}|\mathbf{H}|^{1/2}} \\ &\quad + \mathbf{w}^\top\left(\left(1-\frac{1}{n}\right)\boldsymbol{\Omega}_2 - 2\boldsymbol{\Omega}_1 + \boldsymbol{\Omega}_0\right)\mathbf{w}\,. \end{aligned} \tag{C.6}$$

with

$$\begin{aligned}\mathbb{E}[\hat{f}^2(\boldsymbol{x})] &= \frac{1}{n^2}\sum_i\sum_j \mathbb{E}\left[\phi_{\mathbf{H}}(\boldsymbol{x}-\boldsymbol{x}_i)\phi_{\mathbf{H}}(\boldsymbol{x}-\boldsymbol{x}_j)\right] \\ &= \frac{1}{n^2}\sum_{i=j}\mathbb{E}\left[\phi^2_{\mathbf{H}}(\boldsymbol{x}-\boldsymbol{x}_i)\right] \\ &\quad + \frac{1}{n^2}\sum_{i\neq j}\mathbb{E}\left[\phi_{\mathbf{H}}(\boldsymbol{x}-\boldsymbol{x}_i)\phi_{\mathbf{H}}(\boldsymbol{x}-\boldsymbol{x}_j)\right] \\ &= \frac{1}{n}\mathbb{E}\left[\phi^2_{\mathbf{H}}(\boldsymbol{x}-\boldsymbol{x}_i)\right] \\ &\quad + \frac{n^2-n}{n^2}\mathbb{E}^2\left[\phi_{\mathbf{H}}(\boldsymbol{x}-\boldsymbol{x}_i)\right] \\ &= \frac{1}{n}\frac{1}{(4\pi)^{D/2}|\mathbf{H}|^{1/2}} + \left(1-\frac{1}{n}\right)\mathbf{w}^\top\boldsymbol{\Omega}_2\mathbf{w}\,, \end{aligned} \tag{C.7}$$

where the third equality is due to the fact that $\boldsymbol{x}_i$'s are independently drawn from f. Moreover,

$$\begin{aligned}&\mathbb{E}\left[\int f_0(\boldsymbol{x})\hat{f}(\boldsymbol{x})d\boldsymbol{x}\right] \\ &= \frac{1}{n}\mathbb{E}\left[\sum_i\int\phi_{\mathbf{H}_0}(\boldsymbol{x}-\boldsymbol{\mu}_0)\phi_{\mathbf{H}}(\boldsymbol{x}-\boldsymbol{x}_i)d\boldsymbol{x}\right] \\ &= \frac{1}{n}\mathbb{E}\left[\sum_i\phi_{\mathbf{H}+\mathbf{H}_0}(\boldsymbol{x}_i-\boldsymbol{\mu}_0)\right] \\ &= \mathbb{E}\left[\phi_{\mathbf{H}+\mathbf{H}_0}(\boldsymbol{x}-\boldsymbol{\mu}_0)\right] \\ &= \int\sum_\ell w_\ell\phi_{\boldsymbol{\Sigma}_\ell}(\boldsymbol{x}-\boldsymbol{\mu}_\ell)\phi_{\mathbf{H}+\mathbf{H}_0}(\boldsymbol{x}-\boldsymbol{\mu}_0)d\boldsymbol{x} \\ &= \sum_\ell w_\ell\phi_{\mathbf{H}+\mathbf{H}_0+\boldsymbol{\Sigma}_\ell}(\boldsymbol{\mu}_\ell-\boldsymbol{\mu}_0)\,. \end{aligned} \tag{C.8}$$

Interestingly, the probability of $d(f,\hat{f})$ being far from its ensemble average $\mathbb{E}[d(f,\hat{f})]$

can be bounded as [89]

$$\Pr\left(\left|\|f-\hat{f}\|_p - \mathbb{E}\left[\|f-\hat{f}\|_p\right]\right| \geq t\right) \leq 2\exp\left(-\frac{nt^2h^{2-2/p}}{2\|K_{\mathbf{I}}(\boldsymbol{x})\|_p^2}\right), \tag{C.9}$$

where $\|\cdot\|_p$ denotes the ℓ_p-norm for $p \geq 1$, i.e., $\|f\|_p := \left(\int |f(x)|^p dx\right)^{1/p}$. For the Gaussian kernel with covariance matrix $\mathbf{H} := h^2\mathbf{I}_D$ and $p = 2$, the norm $\|K_{\mathbf{I}}(\boldsymbol{x})\|_p^2$ becomes

$$\begin{aligned}\|K_{\mathbf{I}}(\boldsymbol{x})\|_2^2 &= \|\phi_{\mathbf{I}}(\boldsymbol{x})\|_2^2 \\ &= \int \phi_{\mathbf{I}}^2(\boldsymbol{x})d\boldsymbol{x} = \phi_{2\mathbf{I}}(\mathbf{0}) = \frac{1}{(4\pi)^{D/2}}.\end{aligned} \tag{C.10}$$

Consequently, (C.9) becomes

$$\Pr\left(\left|d(f,\hat{f}) - \mathbb{E}\left[d(f,\hat{f})\right]\right| \geq t\right) \leq 2\exp\left(-\frac{nt^2h\,(4\pi)^{D/2}}{2}\right). \tag{C.11}$$

Letting $q := \Pr(|d(f,\hat{f}) - \mathbb{E}[d(f,\hat{f})]| \geq t)$, (C.11) yields

$$q \leq 2\exp\left(-\frac{nt^2h\,(4\pi)^{D/2}}{2}\right), \tag{C.12}$$

and solving (C.12) w.r.t. t, it is easy to arrive at

$$t \leq \sqrt{-\frac{2\log(q/2)}{nh\,(4\pi)^{D/2}}}. \tag{C.13}$$

Since $1 - q = \Pr(|d(f,\hat{f}) - \mathbb{E}[d(f,\hat{f})]| < t)$, the distance $d(f,\hat{f})$ can be upper bounded from above with probability $1-q$ as

$$\begin{aligned}d(f,\hat{f}) &< \sqrt{-\frac{2\log(q/2)}{nh\,(4\pi)^{D/2}}} + \mathbb{E}[d(f,\hat{f})] \\ &\leq \sqrt{-\frac{2\log(q/2)}{nh\,(4\pi)^{D/2}}} + \sqrt{\mathbb{E}[d^2(f,\hat{f})]}.\end{aligned} \tag{C.14}$$

where the second inequality follows readily from (C.2).

An upper bound with probability $1-q$ to obtain the value of δ_0 can be now derived using (C.3) and (C.14):

$$\begin{aligned}\delta_0 &\leq d(f,\hat{f}) + \delta' \\ &\leq \sqrt{-\frac{2\log(q/2)}{nh\,(4\pi)^{D/2}}} + \sqrt{\mathbb{E}[d^2(f,\hat{f})]} + \delta' := \theta.\end{aligned} \tag{C.15}$$

In lieu of δ_0, θ will be used so as to avoid pathological cases where $1 - \mathbb{E}\left[d(\hat{f}, f_0)\right]/\delta_0 \leq 0$.

Thus using (C.2) and (C.15), the "bad" event probability $\Pr(\mathcal{B}_\delta)$ [cf. (3.26)] can be further lower-bounded with probability $1 - q$ by

$$1 - \frac{\mathbb{E}\left[d(\hat{f}, f_0)\right]}{\theta} \geq 1 - \frac{\sqrt{\mathbb{E}\left[d^2(\hat{f}, f_0)\right]}}{\delta' + \sqrt{-\frac{2\log(q/2)}{nh(4\pi)^{D/2}}} + \sqrt{\mathbb{E}[d^2(f, \hat{f})]}}. \tag{C.16}$$

Finally, using (C.16), the lower bound on $R_{\max}$ of (3.19) reduces to

$$R_{\max} \geq \frac{\log(1-p)}{\log\left(1 - \frac{\sqrt{\mathbb{E}\left[d^2(\hat{f}, f_0)\right]}}{\delta' + \sqrt{-\frac{2\log(q/2)}{nh(4\pi)^{D/2}}} + \sqrt{\mathbb{E}[d^2(f, \hat{f})]}}\right)}, \tag{C.17}$$

with $\zeta := (\mathbb{E}[d^2(\hat{f}, f_0)])^{1/2}$, $\theta_1 := (-2\log(q/2)/(nh(4\pi)^{D/2}))^{1/2} + (\mathbb{E}[d^2(f, \hat{f})])^{1/2}$, and $\theta_2 := \delta'$, that establish the bound of Thm. 1.2.