LARGE-SCALE HIGH-DIMENSIONAL CLUSTERING WITH FAST SKETCHING

Antoine Chatalic⋆*, Rémi Gribonval*† *and Nicolas Keriven*⋆

⋆ Université de Rennes 1, France † Inria Rennes, France

ABSTRACT

In this paper, we address the problem of high-dimensional k-means clustering in a large-scale setting, i.e. for datasets that comprise a large number of items. Sketching techniques have already been used to deal with this "large-scale" issue, by compressing the whole dataset into a single vector of random nonlinear generalized moments from which the k centroids are then retrieved efficiently. However, this approach usually scales quadratically with the dimension; to cope with high-dimensional datasets, we show how to use fast structured random matrices to compute the sketching operator efficiently. This yields significant speed-ups and memory savings for high-dimensional data, while the clustering results are shown to be much more stable, both on artificial and real datasets.

Index Terms— Sketching, Sketched Learning, Fast Transforms, Structured Matrices, k-means, Random Fourier Features.

1. INTRODUCTION

Let $\mathcal{X} = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ be a set of n d-dimensional points. We consider the problem of k-means clustering, which consists in finding k centroids $\mathcal{C} = \{c_1, \dots, c_k\} \subset \mathbb{R}^d$ minimizing the sum of squared errors (SSE):

$$\mathrm{SSE}(\mathcal{X}, \mathcal{C}) = \textstyle\sum_{i=1}^n \min_j \|x_i - c_j\|^2 . \quad (1)$$

We are interested in the case where the size of the dataset n, the dimension d, and possibly the number of clusters k are large. The standard iterative k-means heuristic of Lloyd [1] is widely used because of its simplicity but scales in $\Theta(ndk)$ per iteration, cannot be easily distributed, and requires to load all the dataset in memory, which limits its usability in this context.

A framework [2] has been proposed to deal with large collections by compressing the whole dataset into a single m-dimensional vector $\hat{z}$ of random generalized moments calculated as follows:

$$\hat{z} = \tfrac{1}{n}\textstyle\sum_{i=1}^n \Phi(x_i), \text{ where } \Phi(x) = [e^{-i\omega_1^T x}, \dots, e^{-i\omega_m^T x}]^T. \quad (2)$$

The ω_i are here frequency vectors drawn i.i.d. from an isotropic distribution Λ [3]. The sketch is therefore simply made of m random samples of the empirical characteristic function. Note that the sketching process can be very easily performed in a distributed manner, or even on a data stream. As depicted in Figure 1, the centroids $\mathcal{C}$ and associated weights α can then be estimated efficiently from this sketch, and without using the initial data, as one solution of:

$$\mathcal{C}, \alpha \in \arg\min_{\mathcal{C},\alpha} \left\| \hat{z} - \textstyle\sum_{i=1}^k \alpha_i \Phi(c_i) \right\|_2 . \quad (3)$$

This problem has many similarities with compressive sensing, and theoretical guarantees have been obtained in this context [4].

The optimization problem is non-convex, but approximate solutions can be found using greedy heuristics such as CL-OMPR [3] which is inspired from orthogonal matching pursuit. Some other algorithms, bearing similarities with CL-OMPR, have been proposed in other contexts for solving such sparse inverse problems [5, 6].

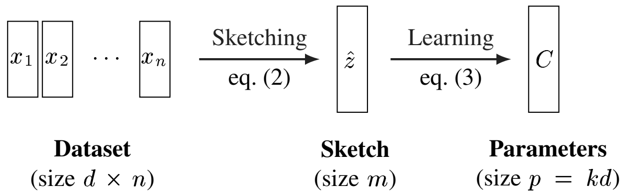


Fig. 1: Overview of the general workflow. In practice, the sketch size m should be of the order of p to get good results, and $p = kd$.

If $X = [x_1, \dots, x_n]$ denotes the dataset in a matrix form, and $W = [\omega_1, \dots, \omega_m]$ the dense $d \times m$ matrix of frequency vectors, computing the sketch involves computing the matrix product $W^T X$. Previous empirical studies [2] suggested that the size m of the sketch should be of the order of the number of parameters to learn (i.e. $m \approx p = kd$) to obtain a quality of clustering that is similar to k-means. Under this assumption, the cost of both sketching and learning phases is dominated by such matrix products and scales quadratically with d. Multiplications by large random matrices appear in various contexts, and multiple works have proposed to replace them by random structured matrices, which behave similarly but have less degrees of freedom, and for which the matrix product can be computed efficiently.

We show how to combine in a single framework the scalability of the sketching approach — thus allowing us to cope with very large and distributed collections (large n) — with the computational efficiency of such fast transforms (large d). In our approach, both sketching and learning phases now scale with d in $\Theta(d \log_2(d))$, which makes it possible to work with high-dimensional data. When the number of centroids k is large as well, we propose to leverage a hierarchical learning algorithm previously introduced for Gaussian mixtures [3], thus reducing the complexity of the learning phase with respect to k from $\Theta(k^3)$ to $\Theta(k^2 \log k)$.

We present some related works in Section 2, detail the proposed method (Section 3) and give experimental results (Section 4); the fast transforms are shown to give significant speed-ups in high dimension, and perhaps surprisingly much more *stable* clustering results both on artificial and real datasets.

2. RELATED WORK

Multiple approaches have been used for large-scale high-dimensional k-means. The dimension of the data can be reduced using feature selection [7], sparsification [8], or geometry-preserving dimensionality-reduction techniques [9] according to the Johnson-Lindenstrauss lemma [10]. Coresets methods [11], on the other side, reduce the size of the dataset but not the dimension. Sketching using random Fourier sampling [12] has been used not only for k-means [2], but also for Gaussian mixtures estimation [3]. The method is reminiscent of the generalized method of moments [13, 14].

Multiplications by large random matrices are used in many contexts; applications include random feature maps computation [12] and cross-polytope locality sensitive hashing. A seminal work in this context is the Fastfood transform [15], which is also included in the broader framework of Choromanski and Sindhwani [16]. In this paper, we use a transformation that is very close to the one of Yu et al [17], and appears in multiple variants in the literature [18, 19], using combinations of diagonal and Walsh-Hadamard matrices as initially proposed by Ailon and Chazelle [20].

3. LEVERAGING FAST TRANSFORMS

We present here a general idea of how to use structured matrices (Section 3.1), show how to build a fast square matrix in the case $d = 2^q$ (Section 3.2) and explain how to generalize in Section 3.3; we propose in Section 3.4 to use a hierarchical learning approach to deal with large values of k, and give a summary of the different methods in Section 3.5.

3.1. General Idea

We explained in the introduction that the sketch (2) can be computed by calculating the matrix product $W^T X$. For large high-dimensional collections, it is unlikely that the amount of available memory will be sufficient, but X can still be divided into multiple chunks if needed. In the extreme case, one can sketch the datapoints (i.e. compute the $\Phi(x_i)$) one at a time and average these sketches on the fly; this scenario is interesting for streaming data as well.

The main contribution of this paper is to demonstrate the benefits of replacing the dense matrix of frequencies W by a fast random structured matrix W_f, such that the sketching operation can be computed efficiently. As the $(\omega_i)_{1\leq i\leq m}$ are drawn i.i.d. from an isotropic distribution, we can rewrite $W = GR$, where R is a $m \times m$ diagonal matrix controlling the distribution of radiuses, and G a Gaussian matrix (i.e. all entries drawn i.i.d. from $\mathcal{N}(0, 1)$), which will be easier to "approximate" with structured matrices. In the following, we use a construction of the form $W_f = G_f R_f$, where R_f is similar to R up to renormalization, and G_f is a fast ersatz of a Gaussian matrix.

3.2. Structured Square Matrices

We explain here how to build a single structured square block (which is similar to assuming $m = d$), when $d = 2^q$. The construction of the structured matrix W_f from these square blocks is detailed just after. The fast transform that we use relies on Walsh-Hadamard matrices, which are defined recursively as follows (still for $d = 2^q$):

$$H_1 = [1] \text{ and } H_{2d} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_d & H_d \\ H_d & -H_d \end{bmatrix}, \quad (4)$$

where H_d is a matrix of dimension $d \times d$. The product $H_d x$ can be computed using the fast Walsh-Hadamard transform, whose complexity scales in $\Theta(d \log(d))$, rather than using the standard $\Theta(d^2)$ matrix-vector product.

In the following, H always denotes a Walsh-Hadamard matrix, and we consider diagonal matrices D_i with diagonal entries drawn independently using the Rademacher distribution, i.e. ± 1 with probability $1/2$. Ailon and Chazelle have initially introduced this block HD as a preprocessing step in order to smooth the energy distribution of input data vectors before applying a sparse transform [20].

	CKM	**FCKM**	KM
Time	$\boldsymbol{kd^2(n+k^2)}$	$\boldsymbol{kd\ln(d)(n+k\ln(k))}$	$\boldsymbol{ndkI}$
Sketching	nkd^2	$nkd\ln(d)$	n/a
Learning			
CL-OMPR	k^3d^2	$k^3d\ln(d)$	n/a
Hierarchical	$k^2\ln(k)d^2$	$k^2\ln(k)d\ln(d)$	n/a
Space	$\boldsymbol{kd(d+n_b)}$	$\boldsymbol{kdn_b}$	$\boldsymbol{nd}$
Sketch	kd	kd	n/a
W	kd^2	kd	n/a
W^TX	kdn_b	kdn_b	n/a

Table 1: Time and space costs for KM, CKM and FCKM, assuming $m \approx kd$. Here d denotes the dimension, n the size of the collection, n_b the batch size for sketching, k the number of clusters, I the number of iterations. All complexities should be read as $\Theta(\cdot)$.

Now for $d = 2^q$, a square block B can be built as:

$$B = \underset{R}{\boxed{\begin{matrix} r_1 & & 0 \\ & \ddots & \\ 0 & & r_d \end{matrix}}} \; \underset{H}{\boxed{H_d}} \; \underset{D_1}{\boxed{\begin{matrix} \pm1 & & 0 \\ & \ddots & \\ 0 & & \pm1 \end{matrix}}} \; \underset{H}{\boxed{H_d}} \; \underset{D_2}{\boxed{\begin{matrix} \pm1 & & 0 \\ & \ddots & \\ 0 & & \pm1 \end{matrix}}} \; \underset{H}{\boxed{H_d}} \; \underset{D_3}{\boxed{\begin{matrix} \pm1 & & 0 \\ & \ddots & \\ 0 & & \pm1 \end{matrix}}} \quad (5)$$

where the $(D_i)_{1\leq i\leq 3}$ are drawn independently, and the $(r_i)_{1\leq i\leq d}$ i.i.d. from the desired radius distribution. Empirical experiments suggested that using two HD_i blocks was not sufficient, and the first theoretical guarantees have been proposed using three blocks [17].

3.3. Generalization

We assumed previously that d was a power of 2, and explained how to build one square matrix. In the general case, it has been empirically observed [2] that a choice of $m \approx kd$ for the targeted sketch size yields good clustering results, so we need to generalize our construction to arbitrary $m \times d$ matrices with $m > d$, and to deal with dimensions d that are not powers of 2.

When d is not a power of 2, we denote $q = \lceil \log_2(d) \rceil$, $r = \lceil m/2^q \rceil$, $d_{pad} = 2^q$ and $m_{pad} = r2^q$. We use for sketching a $d_{pad} \times m_{pad}$ matrix W_f, whose transpose W_f^T is built by vertically stacking square blocks of size $2^q \times 2^q$ that are drawn independently according to (5). We use zero-padding on the data to get a matrix X_{pad} of matching dimensions, and keep only the m first rows when computing the product $W_f^T X_{pad}$.

The storage cost for this construction is $4rd_{pad} = 4m_{pad}$, which may yield significant savings in high dimension. The cost of sketching is now $\Theta(nkd\log(d))$; the CL-OMPR algorithm, which also involves computing the function Φ as it appears in loss function (3), scales with d in $\Theta(d\log(d))$ as well.

3.4. Hierarchical Learning

Although the learning phase benefits from using fast transforms as well when the dimension d is high, a greedy algorithm such as CL-OMPR also scales with the number of clusters k in $\Theta(k^3)$, which can become prohibitive for some applications. A hierarchical approach has been introduced for learning Gaussian mixtures [3], but never been adapted for k-means; it consists in learning a mixture of Gaussians with diagonal covariance by recursively splitting each Gaussian in two along the dimension of highest variance. We propose to apply this algorithm, and then simply use the centers of the Gaussians as an initialization for minimizing the loss function (3).

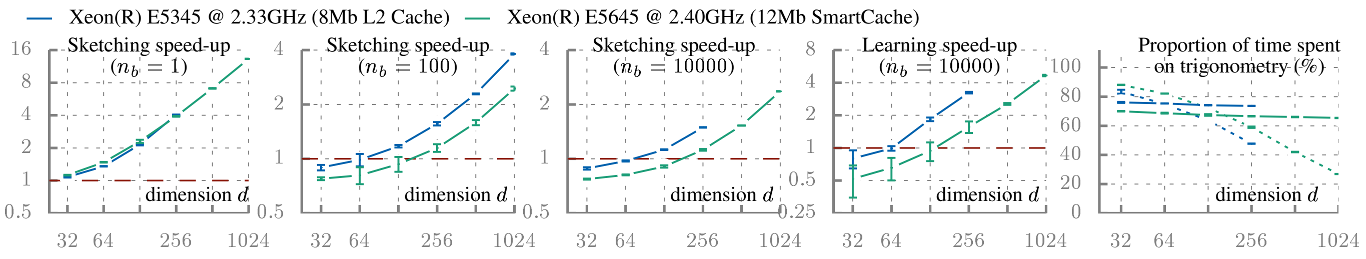


Fig. 2: Sketching speed-ups (i.e. ratios of running times without/with fast transforms) for three values of the batch size n_b, speed-up of the learning phase, and proportion of the sketching time spent on computing the nonlinearity (sines and cosines). Results obtained on 30 experiments. In the rightmost figure, plain and dashed lines denote respectively the usage of fast and dense matrices.

3.5. Summary

In the following, KM stands for "k-means" [1], CKM for "Compressive k-means" [2], and FCKM for "Fast Compressive k-means" — i.e. using structured matrices. We refer to the learning algorithm using fast hierarchical initialization as "Hierarchical", and both the CL-OMPR and Hierarchical algorithms can be used with dense or structured matrices, i.e. in the CKM or FCKM frameworks.

We give in Table 1 a summary of space and time complexities of the different methods assuming $m = \Theta(kd)$, as it empirically [2] seems to be a necessary condition to get good clustering results. One should keep in mind that the sketching time can be drastically reduced by relying on distributed computing, which is not the case when using Lloyd's k-means.

4. EXPERIMENTAL VALIDATION

We first give some implementation details (Section 4.1), and then present experiments on randomly generated (Section 4.2) and real data (Sections 4.3 and 4.4).

4.1. Implementation details

We implemented the fast transform proposed in Section 3 as a contribution to the SketchMLbox Matlab Toolbox [21], which already includes the sketching procedure, the CL-OMPR heuristic and the Hierarchical algorithm for Gaussian mixtures, but always using a dense matrix of frequency vectors.

The critical part of the code is written in C and compiled to binary MEX files. We use the adaptive Fast Walsh-Hadamard Transform of the Spiral project [22, 23], which is designed for an optimized usage of the cache hierarchy [24]. One might get higher speed-ups by relying on carefully designed SIMD implementations [25]. In the experiments, KM always refer to the Matlab implementation of k-means using $I_{max} = 1000$ as the maximum number of iterations, and with uniform initialization — except when using k-means++ in Section 4.3.

4.2. Synthetic Data

We first show on artificial data that one can replace the dense matrix by a structured one without degrading the quality of the results. We perform here k-means clustering on $n = 10000$ data vectors randomly generated according to a mixture of $k = 10$ Gaussians with identity covariance matrix. The means are drawn with respect to a centered Gaussian with covariance $1.5k^{1/d}I_d$ to create clusters that are well separated with high probability [3].

The quality of the clustering is measured using SSE (1), and Figure 3 shows the ratios obtained using CKM or FCKM with respect to Matlab's k-means (KM). These results confirm that the compressive

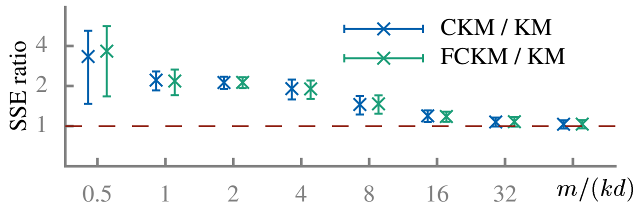


Fig. 3: Ratios of SSE for CKM and FCKM with respect to one run of Matlab's k-means (KM) as a function of $m/(kd)$. Averaged on 30 repetitions for each $d \in \{8, 16, 32, 64, 128, 256, 512\}$, and $k = 10$.

k-means framework gives good clustering results provided that the sketch is large enough, but also that in this case, using fast transforms does not degrade the clustering quality.

Sketching speed-ups obtained using structured rather than dense matrices of frequencies are given in Figure 2. We consider different values of the size n_b of the batches used for sketching — i.e. we compute products such as $W^T X_i$, where X_i is a batch of size $d \times n_b$. The speed-ups are significant for high-dimensional data, and especially in the "streaming" case, i.e. when sketching the vectors one by one; this is very interesting when the amount of available memory is limited. For larger batch sizes, using fast transforms might not be useful for small dimensions (the BLAS matrix-matrix product being too optimized to compete with it), but still allows us to deal with high-dimensional datasets. The learning phase benefits similarly from using fast transforms as depicted on the 4th sub-figure. Note that the computational cost of the complex exponential is very high (5th sub-figure): approximately 70% of the time for all dimensions when using fast transforms. Working with alternative feature maps relying on cheaper non-linear functions could be interesting for future works.

4.3. Spectral clustering on MNIST

As long as we work on high-dimensional datasets, one should expect to observe on real datasets the same speed-ups as the ones obtained on random data. In the following, we consider datasets with dimensions for which one should not expect to get significant speed-ups when using a large batch size according to Figure 2; however, it is highly interesting to check whether the fact that structured matrices yield the same clustering quality on random data (see Section 4.2) also holds for real datasets.

We perform here clustering on the MNIST dataset [26] of handwritten digits, which has $k = 10$ classes. The original dataset contains $n = 7 \times 10^4$ pictures. Distorted variants of these images have been generated using infiMNIST [27], so that one other dataset of size $n = 3 \times 10^5$ is used for evaluation as well. For every image, we extract dense SIFT descriptors [28], which are concatenated into a single vector. We compute the similarity matrix between these vectors, and the k first eigenvectors of the associated Laplacian matrix

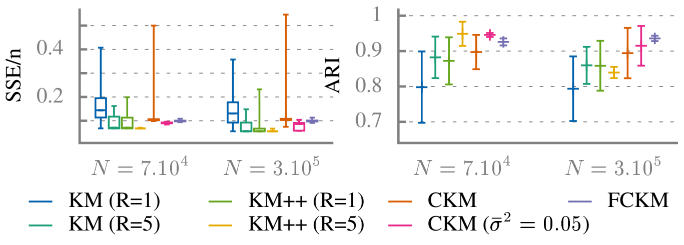


Fig. 4: Min/quartiles/max boxplots and median of SSE (left, the lower the better) and mean & standard deviation of the adjusted Rand index (right, the higher the better). Results obtained on 120 experiments, $m = 1008$; R corresponds to the number of replicates. KM was run with a maximum number of 1000 iterations, using uniform initialization. The legend, when read column by column from KM to FCKM, corresponds to the order of the boxplots from left to right.

in order to get n spectral features [29] in dimension $d = k = 10$.

Figure 4 gives the results in terms of SSE and adjusted Rand index (ARI), for KM, CKM and FCKM. We also consider k-means++ (KM++) [30], and include results with $R = 5$ replicates, i.e. running the algorithm 5 times and keeping the best results. Note that for CKM and FCKM, the distribution used to draw the frequency vectors involves a parameter $\bar{\sigma}^2$ that is estimated automatically by first sketching a small subset of the data [3]. For comparison, we also consider using CKM with a fixed value of $\bar{\sigma}^2$ rather than relying on this estimation. Results obtained with CKM contain roughly 15% of outliers. Using CKM with $\bar{\sigma}^2 = 0.05$ gives better and highly concentrated results, but one usually does not have this knowledge of the dataset. FCKM turns out to give similar results in terms of SSE and ARI, but without requiring any knowledge on $\bar{\sigma}^2$, as it is here again automatically estimated. Results are well concentrated, contrarily to what is obtained with Lloyd's k-means, even with $R = 5$ replicates. KM++ with $R = 5$ replicates gives good results in terms of SSE, but lower ARI for $n = 3 \times 10^5$; one should keep in mind that KM and KM++ require to have the whole dataset in memory.

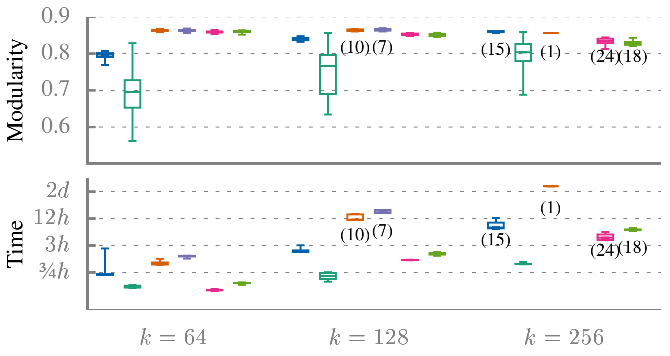
In summary, even in small dimensions where sketching might be faster with dense matrices, the use of structured random matrices is potentially useful in terms of stability. To get the best possible speed, one can rely on the explicit dense representation of such structured matrices if applicable.

4.4. Hierarchical clustering on a co-purchasing graph

Computing spectral features involves computing the eigendecomposition of the Laplacian matrix, which is usually expensive. Tremblay et al. proposed to bypass this step by using as features $\Theta(\log k)$ filtered random graph signals on the graph associated to the Laplacian matrix [31]. Standard KM is then applied on a random subset of these features, and interpolated to the whole collection. We combine these fast random features with the FCKM framework, thus allowing us to avoid the subsampling step.

We work on the Amazon co-purchasing network [32], which is a graph comprising $n = 334863$ nodes and $E = 925872$ edges. As there is no ground truth for this dataset, we used $k = 64, 128, 256$. We compare the original spectral clustering (SC), compressive spectral clustering (CSC) [31], and 4 methods using sketching on the same random features: we combine the two types of matrices (dense/structured) with the two learning procedures discussed in Section 3.4 (CL-OMPR/Hierarchical). Please refer to the table of Figure 5 for a summary.

Standard KM is launched with $R = 2$ replicates. Using compressive spectral clustering, the k-means step is performed only on a



	Method	Features	Subs.	Sk. matrix	Clustering
—	SC	spectral	No	n/a	KM
—	CSC	random	Yes	n/a	KM
—	S2C	random	No	Dense	CL-OMPR
—	FS2C	random	No	Structured	CL-OMPR
—	HS2C	random	No	Dense	Hierarchical
—	HFS2C	random	No	Structured	Hierarchical

Fig. 5: Boxplots of modularity (the higher the better) and clustering time for $k = 64, 128, 256$. Only the learning times are displayed for sketching methods; sketching times are much smaller, even on a single core. All experiments were run on Intel(R) Xeon(R) CPUs E5640 and repeated 30 times (or less for experiments that were too long; the number of iterations is indicated below in these cases, and FS2C does not appear for $k = 256$). The table is a summary of the different methods (in the order of the boxplots, from left to right).

subset of the features and is therefore much faster; we used $R = 20$ replicates for a fair comparison. We used $m = 10kd$ for the sketch size when using CKM. All initializations were performed uniformly.

The results are presented in Figure 5 (top), where the elapsed times are given and the clustering quality is measured with the modularity metric [33]. As regards the elapsed times, CL-OMPR is not competitive but satisfying results are obtained with the hierarchical algorithm. Similar modularities are obtained with and without structured matrices; the results are slightly lower when using the hierarchical algorithm, but in both cases they are highly concentrated, whereas CSC yields a high variance.

5. PERSPECTIVES

We proposed a way of combining the efficiency of fast structured matrices with the scalability of sketching-based approaches and hierarchical learning methods, yielding a k-means framework which can handle large and high-dimensional collections with a limited memory footprint. Experimental validation confirms that significant speedups are obtained in high dimension, and the clustering results seem to be much more stable.

It would of course be interesting to be able to control the error induced by the use of structured matrices. In the theoretical framework [4] which has been proposed for sketched learning, the matrix used for sketching implicitly defines a kernel function, and studying the kernel associated with the structured matrices used in our approach should help to establish theoretical guarantees.

Although the hierarchical algorithm proposed for the approximate minimization allows to deal with a larger number of centroids compared to CL-OMPR, it still scales quadratically with k when $m = \Theta(kd)$, leaving room for improvement; locality could for instance be leveraged, as it has been done for orthogonal matching pursuit [34]. Designing provably-good procedures for this optimization problem is challenging as well.

6. REFERENCES

[1] Stuart Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[2] Nicolas Keriven, Nicolas Tremblay, Yann Traonmilin, and Rémi Gribonval, "Compressive k-means," 2017.

[3] Nicolas Keriven, Anthony Bourrier, Rémi Gribonval, and Patrick Pérez, "Sketching for large-scale learning of mixture models," *To be published in Information and Inference*, vol. abs/1606.02838, 2016.

[4] Rémi Gribonval, Gilles Blanchard, Nicolas Keriven, and Yann Traonmilin, "Compressive statistical learning with random feature moments," .

[5] Kristian Bredies and Hanna Katriina Pikkarainen, "Inverse problems in spaces of measures," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 19, no. 1, pp. 190–218, 2013.

[6] Nicholas Boyd, Geoffrey Schiebinger, and Benjamin Recht, "The alternating descent conditional gradient method for sparse inverse problems," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 616–639, 2017.

[7] Christos Boutsidis, Petros Drineas, and Michael W Mahoney, "Unsupervised feature selection for the k-means clustering problem," in *Advances in Neural Information Processing Systems*, 2009, pp. 153–161.

[8] Piotr Indyk, Jiří Matoušek, and Anastasios Sidiropoulos, "Low-distortion embeddings of finite metric spaces," in *Handbook of discrete and computational geometry*, Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman, Eds. CRC press.

[9] Christos Boutsidis, Anastasios Zouzias, and Petros Drineas, "Random projections for k-means clustering," in *Advances in Neural Information Processing Systems*, 2010, pp. 298–306.

[10] William B. Johnson and Joram Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," vol. 26, no. 189.

[11] Jeff M. Phillips, "Coresets and sketches," in *Handbook of discrete and computational geometry*, Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman, Eds., chapter 48. CRC press, 2017.

[12] Ali Rahimi and Benjamin Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems (NIPS)*. 2007, pp. 1177–1184, Curran Associates, Inc.

[13] Alastair R Hall, *Generalized method of moments*, Oxford University Press, 2005.

[14] Lars Peter Hansen, "Large sample properties of generalized method of moments estimators," *Econometrica: Journal of the Econometric Society*, pp. 1029–1054, 1982.

[15] Quoc Le, Tamás Sarlós, and Alex Smola, "Fastfood-approximating kernel expansions in loglinear time," in *Proceedings of the international conference on machine learning (ICML)*, 2013.

[16] Krzysztof Choromanski and Vikas Sindhwani, "Recycling randomness with structure for sublinear time kernel expansions," in *Proceedings of the international conference on machine learning (ICML)*. 2016, vol. 48 of *JMLR Workshop and Conference Proceedings*, pp. 2502–2510, JMLR.org.

[17] Felix X. Yu, Ananda Theertha Suresh, Krzysztof Marcin Choromanski, Daniel Holtmann-Rice, and Sanjiv Kumar, "Orthogonal random features," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 1975–1983.

[18] Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Francois Fagan, Cedric Gouy-Pailler, Anne Morvan, Nouri Sakr, Tamas Sarlos, and Jamal Atif, "Structured adaptive and random spinners for fast machine learning computations," in *The 20th International Conference on Artificial Intelligence and Statistics*.

[19] Krzysztof Choromanski, Mark Rowland, and Adrian Weller, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems (NIPS)*.

[20] Nir Ailon and Bernard Chazelle, "The fast johnson–lindenstrauss transform and approximate nearest neighbors," vol. 39, no. 1, pp. 302–322.

[21] Nicolas Keriven, Nicolas Tremblay, and Rémi Gribonval, "Sketchmlbox: A matlab toolbox for large-scale mixture learning. `http://sketchml.gforge.inria.fr/`," 2016.

[22] "Spiral project: Wht package. `http://www.spiral.net/software/wht.html`," .

[23] J. Johnson and M. Puschel, "In search of the optimal walsh-hadamard transform," in *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, vol. 6.

[24] Neungsoo Park and N. K. Prasanna, "Cache conscious walsh-hadamard transform," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing 2001 (ICASSP)*, vol. 2, pp. 1205–1208 vol.2.

[25] Joachim Curto, Irene Zarza, Feng Yang, Alexander Smola, and Luc Van Gool, "F2f: A library for fast kernel expansions," .

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," vol. 86, no. 11, pp. 2278–2324.

[27] Gaëlle Loosli, Stéphane Canu, and Léon Bottou, "Training invariant support vector machines using selective sampling," *Large scale kernel machines*, pp. 301–320, 2007.

[28] David G Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[29] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, pp. 849–856.

[30] David Arthur and Sergei Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

[31] Nicolas Tremblay, Gilles Puy, Rémi Gribonval, and Pierre Vandergheynst, "Compressive spectral clustering," in *ICML 2016), June*, pp. 20–22.

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

[33] Mark EJ Newman and Michelle Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, pp. 026113, 2004.

[34] Boris Mailhé, Rémi Gribonval, Frédéric Bimbot, and Pierre Vandergheynst, "LocOMP: algorithme localement orthogonal pour l'approximation parcimonieuse rapide de signaux longs sur des dictionnaires locaux," in *GRETSI*.