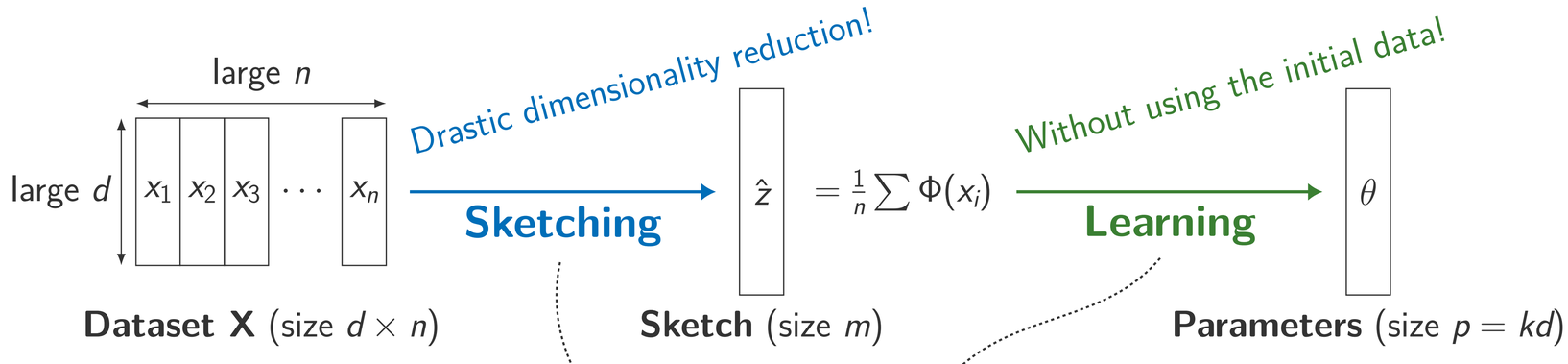
Large-Scale High-Dimensional Clustering with Fast Sketching

Antoine Chatalic (Université de Rennes 1), Rémi Gribonval (Inria Rennes) and Nicolas Keriven (Université de Rennes 1).

Context

How to learn from a **large** collection of **high-dimensional** elements?



Framework

Sketching can be used to learn mixture models from **large collections (large n)** [2]:

1. **Sketching phase:** the whole dataset is compressed **into a single m-dimensional vector** $\hat{z}$ of random generalized moments $\hat{z}$ w.r.t. m frequency vectors $(\omega_i)_{1\leq i\leq m}$:
$$\hat{z} = \frac{1}{n}\sum_{i=1}^{n} \Phi(x_i), \text{ where } \Phi : x \mapsto \left[e^{-i\omega_1^T x}, \dots, e^{-i\omega_m^T x} \right]^T. \quad (1)$$
2. **Learning phase:** The parameters of the mixture components $\mathcal{C}$ are estimated (using only the sketch) by solving:
$$\mathcal{C}, \alpha \in \arg\min_{\mathcal{C},\alpha} \left\| \hat{z} - \sum_{i=1}^{k} \alpha_i \Phi(c_i) \right\|_2. \quad (2)$$

State of the art:

- Optimization (2) is solved using the greedy heuristic CL-OMPR (inspired from orthogonal matching pursuit);
- W is a (rescaled) dense Gaussian matrix of size $m \times d$.
  - Sketching $\rightsquigarrow$ compute $W^T X$ $\rightsquigarrow$ scales in $\Theta(mdn)$
  - Learning $\rightsquigarrow$ compute similar products by W and W^T $\rightsquigarrow$ $\Theta(mdk^2)$!

We propose to replace W by a structured random matrix.

Contribution: sketching with structured matrices

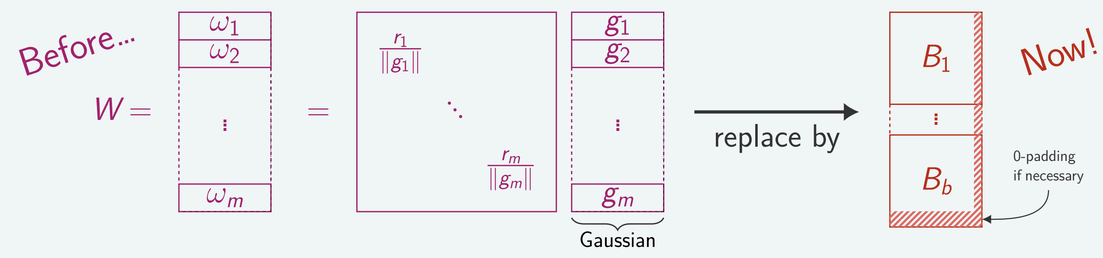
For $d = 2^q$, a "fast" block B of size $d \times d$ an be built with the following structure [1, 4]:

$$B = \begin{bmatrix} r_1 & & & & & & \\ & \ddots & & & & & \\ & & r_d & & & & \end{bmatrix} \begin{bmatrix} & & \\ & \mathrm{H} & \\ & & \end{bmatrix} \begin{bmatrix} \pm 1 & & \\ & \ddots & \\ & & \pm 1 \end{bmatrix} \begin{bmatrix} & & \\ & \mathrm{H} & \\ & & \end{bmatrix} \begin{bmatrix} \pm 1 & & \\ & \ddots & \\ & & \pm 1 \end{bmatrix} \begin{bmatrix} & & \\ & \mathrm{H} & \\ & & \end{bmatrix} \begin{bmatrix} \pm 1 & & \\ & \ddots & \\ & & \pm 1 \end{bmatrix} \quad (3)$$

Radiuses — Hadamard — Hadamard — Hadamard

Fast Walsh-Hadamard transform $\rightsquigarrow$ matrix-vector products cost $\Theta(d\log_2(d))$ only!

We build a structured matrix by stacking b such blocks $(B_i)_i$ drawn i.i.d. according to (3):



Application: k-means clustering

- **Input:** $\mathcal{X} = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ a set of n d-dimensional points.
- **Output:** k centroids $\mathcal{C} = \{c_1, \dots, c_k\} \subset \mathbb{R}^d$ minimizing the sum of squared errors:
$$\mathrm{SSE}(\mathcal{X}, \mathcal{C}) = \sum_{i=1}^{n} \min_j \|x_i - c_j\|^2. \quad (4)$$

We want to learn $p = kd$ parameters; empirically, it turns out that we need $m \approx p = kd$ to get good clustering results.

Summary

- **KM:** k-means (Lloyd's algorithm). Time: $\Theta(ndk)$; Space: $\Theta(nd)$!
- **CKM:** Compressive k-means.
- **FCKM:** Fast compressive k-means.

		Before	Now
Algorithm		*CKM*	*FCKM*
Matrix of frequencies		*Dense*	*Structured*
Time:	Sketching	nkd^2	$nkd\ln(d)$
	Learning → CL-OMPR	k^3d^2	$k^3d\ln(d)$
	→ Hierarchical	$k^2\ln(k)d^2$	$k^2\ln(k)d\ln(d)$
Space:	W	kd^2	kd
	W^TX	kdn_b	kdn_b

Experimental validation

Randomly generated data

Implementation: SketchMLbox toolbox.

- $n = 10^4$, $k = 10$, $d \in \{8, 16, \dots, 512\}$, metric: ratio of SSE.
- $(x_i)_{1\leq i\leq n}$ drawn according to a mixture of k separated Gaussians.

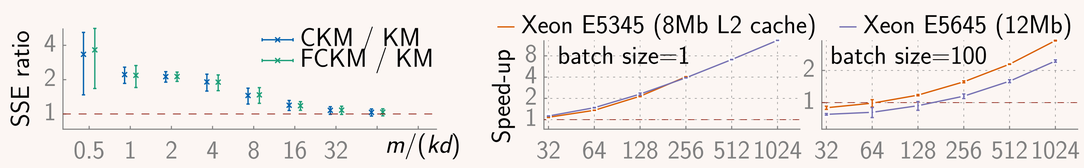


Figure: SSE ratios as a function of m/kd, and sketching speed-ups for 2 different batch sizes.

Conclusions:

- Using fast matrices gives the **same clustering quality.**
- For d not too small, fast transforms give **significant speedups**.

Clustering quality on the MNIST dataset

Spectral clustering $\rightsquigarrow$ $d = k = 10$ $\rightsquigarrow$ no speedup, but what about clustering quality? Metric: SSE and adjusted Rand index (ARI).

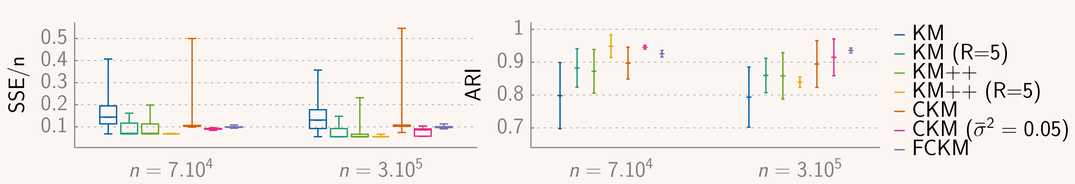


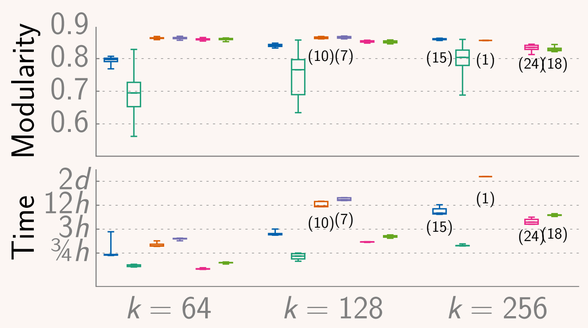
Figure: SSE (the lower the better) & ARI (the higher the better); $m = 1008$; 120 exps; R="replicates"; uniform initialization.

Conclusions:

- Using fast transforms gives **similar or slightly better results.**
- Results are **more stable using fast transforms** $\rightsquigarrow$ interesting even for small d.

Hierarchical clustering on Amazon co-purchasing graph

We work on an Amazon co-purchasing network, ($n \approx 3.10^5$ nodes), using spectral and random features [3]. The clustering quality is measured with modularity.



Features	Subs.	W matrix	Algorithm
spectral	No	n/a	KM
random	Yes	n/a	KM
random	No	Dense	CL-OMPR
random	No	Structured	CL-OMPR
random	No	Dense	Hierarchical
random	No	Structured	Hierarchical

Figure: Boxplots of modularity (the higher the better) and clustering time. Using Xeons E5640, 30 repetitions, $R = 2$ for KM, $R = 20$ for KM+subsampling, $m = 10kd$.

Conclusions:

- The same clustering quality is obtained when using fast transforms.
- The hierarchical algorithm is **much faster**, and achieves similar modularities.

What's next?

- Extend this framework to other learning tasks.
- Design more efficient algorithms to solve the optimization problem (2).

References

[1] Nir Ailon and Bernard Chazelle. "The fast Johnson–Lindenstrauss transform and approximate nearest neighbors". 2009.
[2] Nicolas Keriven et al. "Compressive K-means". 2017.
[3] Nicolas Tremblay et al. "Compressive spectral clustering". 2016.
[4] Felix X. Yu et al. "Orthogonal Random Features". 2016.