PLEASE

projection, learning and sparsity for efficient data processing

Learning from random moments

Rémi Gribonval - Inria Rennes - Bretagne Atlantique

remi.gribonval@inria.fr

Joint work with: G. Blanchard (U. Potsdam)

N. Keriven, Y Traonmilin (Inria Rennes)

Main Contributors & Collaborators

- Anthony Bourrier

- Nicolas Keriven

- Yann Traonmilin

- Gilles Puy

- Nicolas Tremblay

- Gilles Blanchard

- Mike Davies

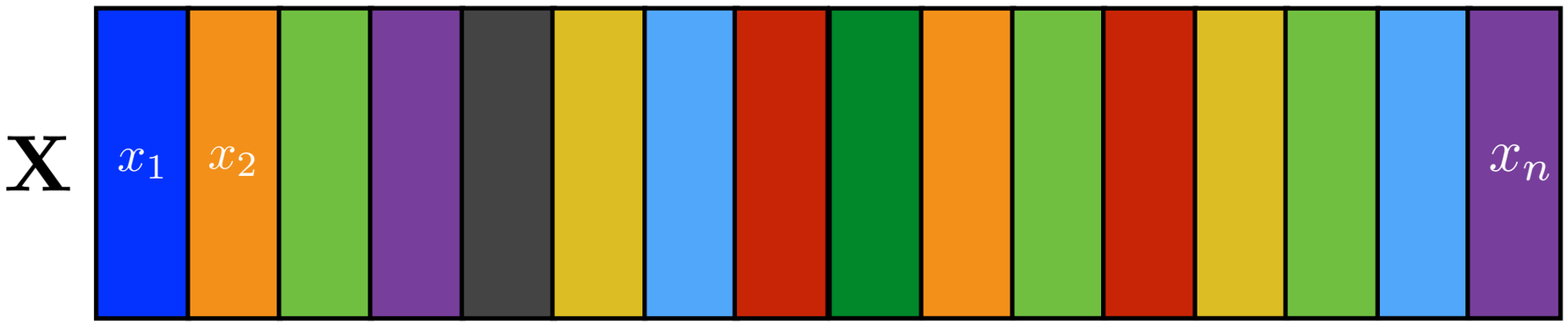
- Patrick Perez

Foreword

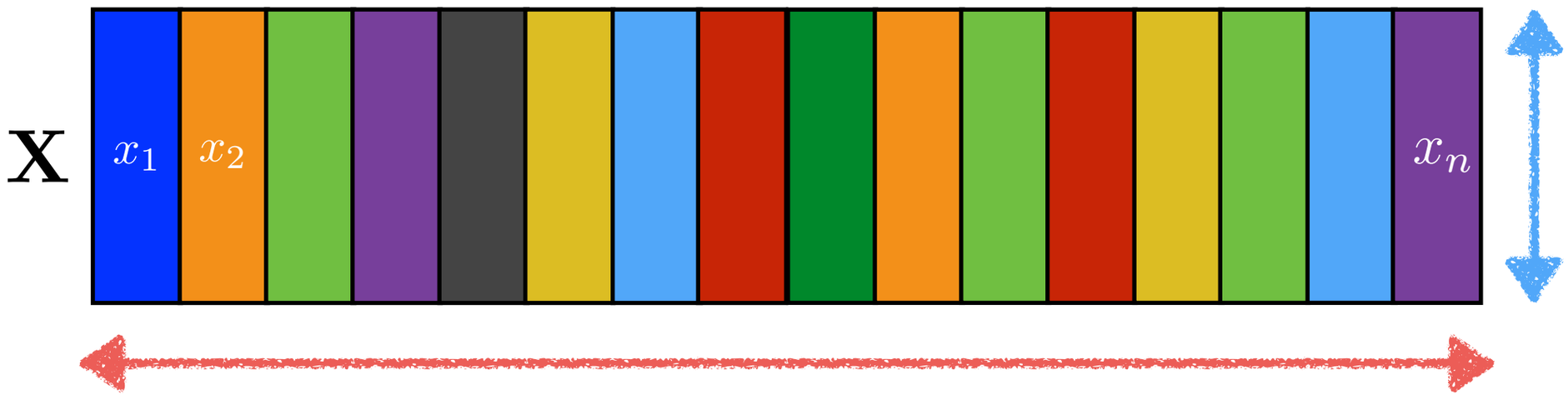
- Signal processing & *machine learning*
  - inverse problems & *generalized method of moments*
  - embeddings with random projections & *random features /kernels*
  - image super-resolution, source localization & *k-means*
- Continuous *vs* discrete ?
  - wavelets (1990s): from continuous to discrete
  - compressive sensing (2000s): in the discrete world
  - current trends : back to continuous !
    - off-the-grid compressive sensing, FRI, high-resolution methods
    - *compressive statistical learning from random moments*

- Learning from random moments: the concept
- Compressive *Statistical* Learning (guarantees)
- Recent developments & perspectives

Large-scale learning

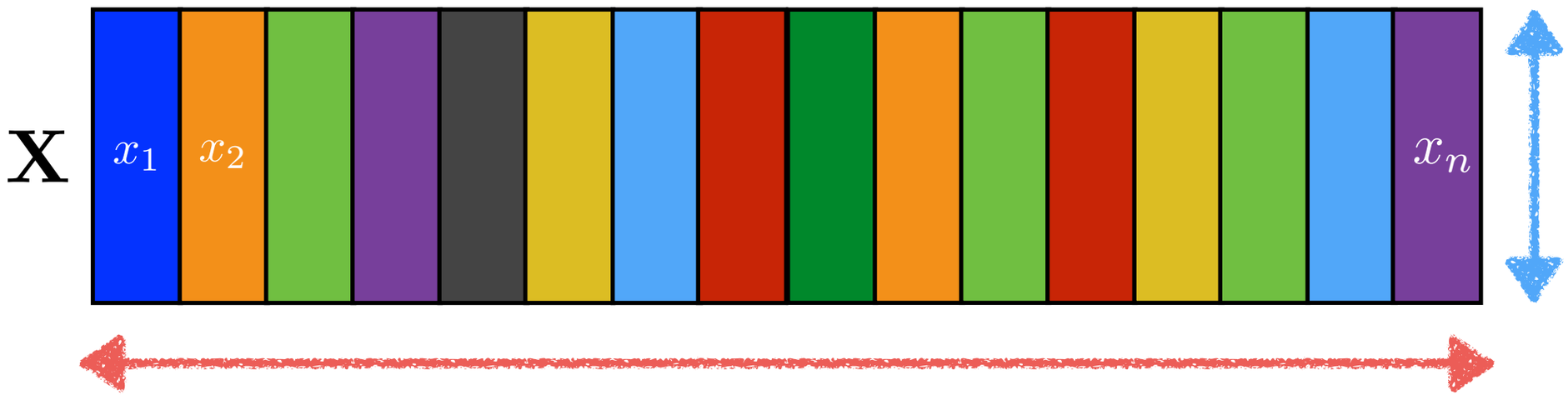


Large-scale learning



- High feature dimension *d*
- Large collection size *n* = *“volume”*

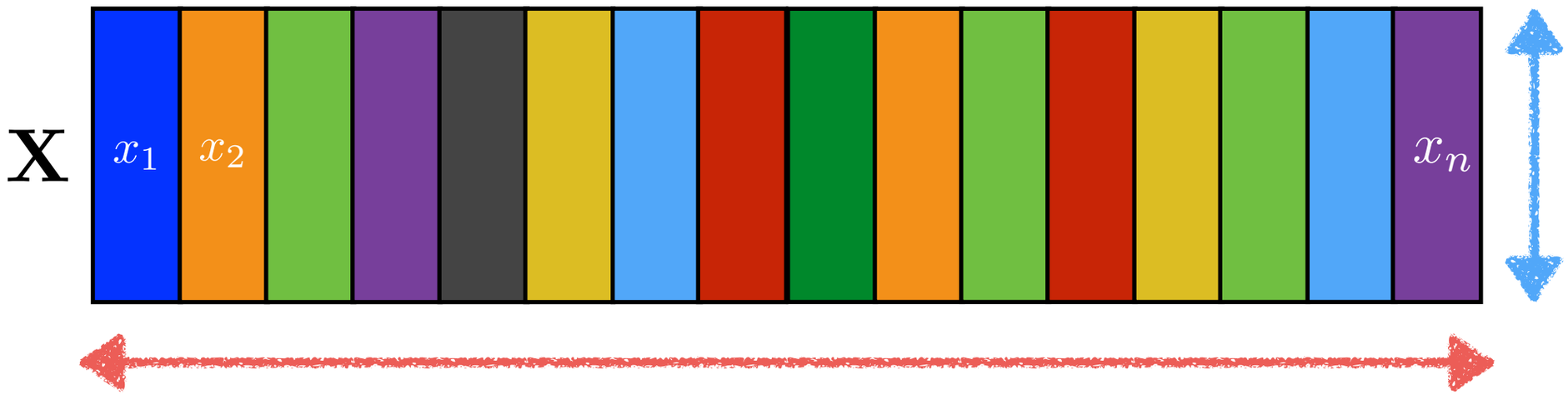
Large-scale learning



- High feature dimension *d*
- Large collection size *n* = *"volume"*

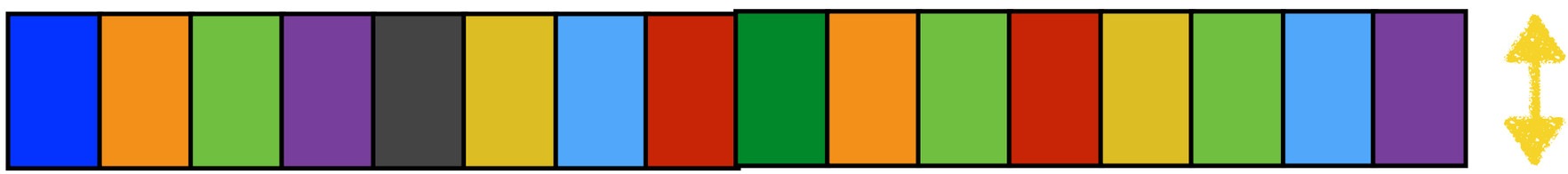
Challenge: compress $\mathcal{X}$ before learning ?

Compressive learning: three routes



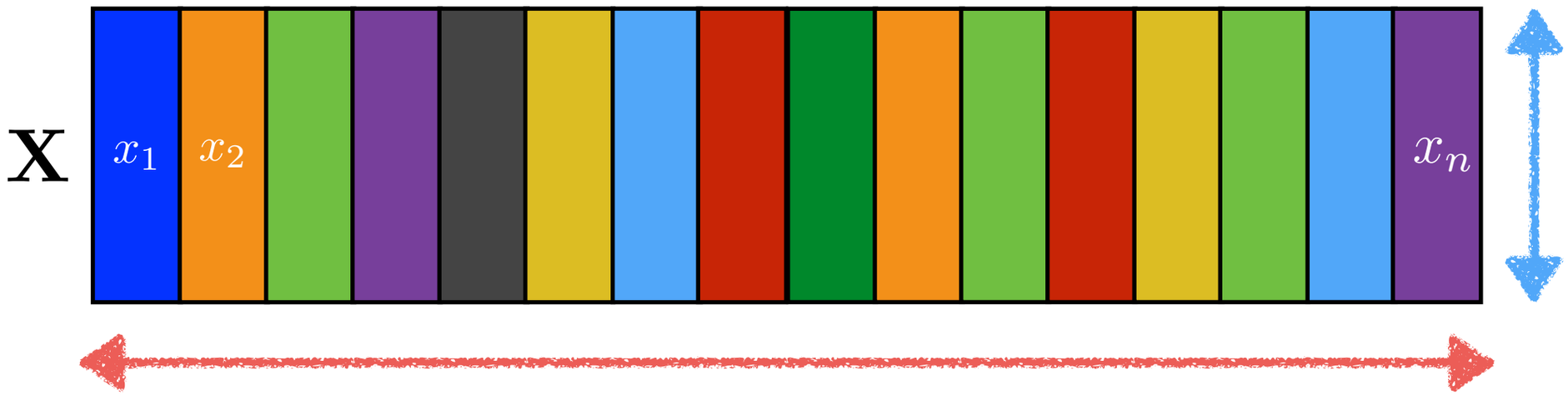
- dimension reduction
- subsampling
- **sketching**

$$\mathbf{Y} = \mathbf{M}\mathbf{X}$$

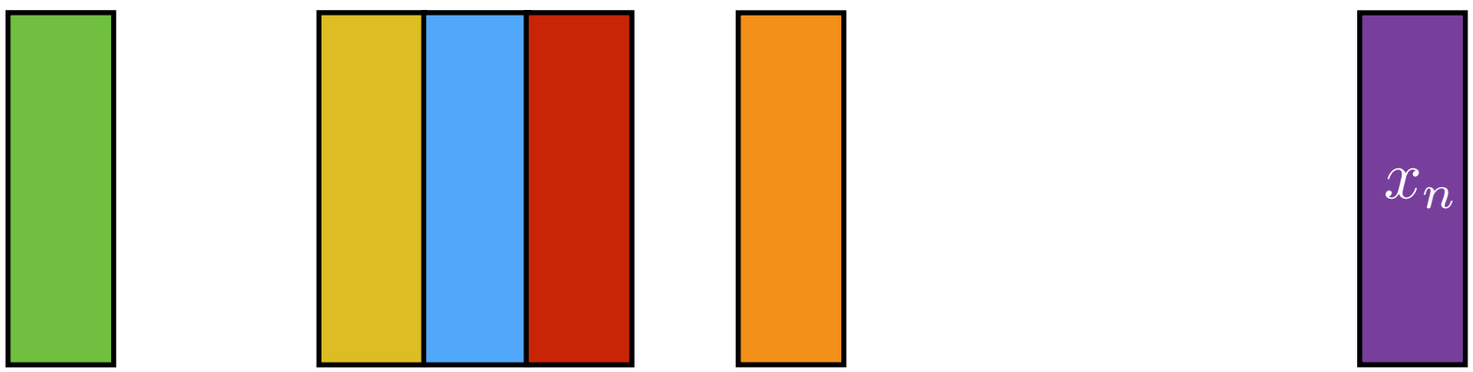


random projections - Johnson Lindenstrauss lemma
see e.g. [Calderbank & al 2009, Reboredo & al 2013]

Compressive learning: three routes



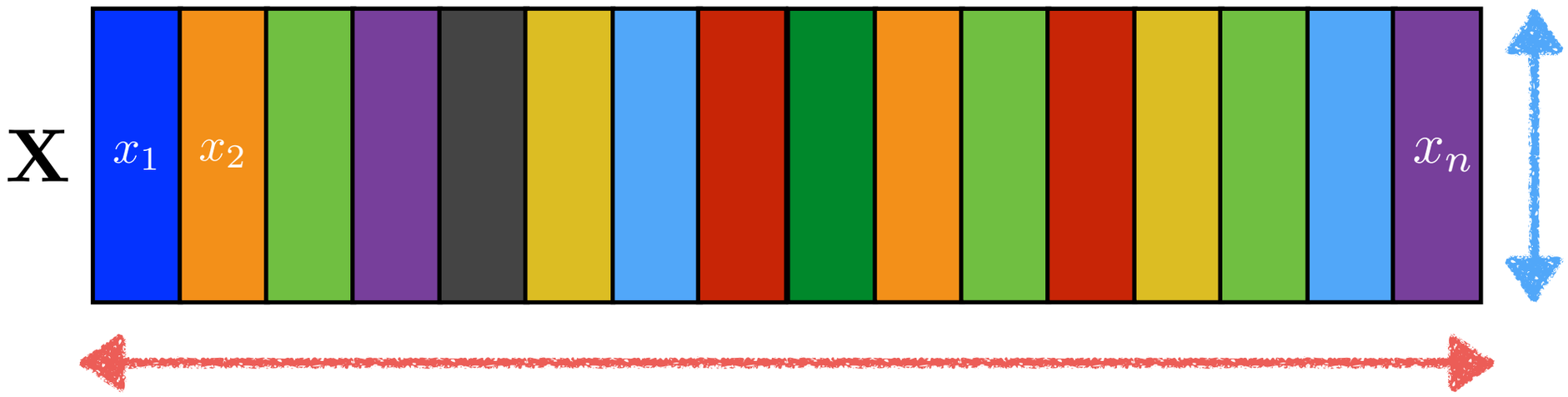
- dimension reduction
- subsampling
- **sketching**



Nyström method & coresets

see e.g. [Williams&Seeger 2000, Agarwal & al 2003, Felman 2010]

Compressive learning: three routes



- dimension reduction
- subsampling
- **random moments**

$$\begin{matrix}\mathbb{E}\Phi_1(X)\\ \dots \\ \mathbb{E}\Phi_m(X)\end{matrix} \quad z \in \mathbb{R}^m$$

Inspiration: *compressive sensing* *[Foucart & Rauhut 2013]*
sketching/hashing *[Thaper & al 2002, Cormode & al 2005]*
Connections with: *generalized method of moments* *[Hall 2005]*
kernel mean embeddings *[Smola & al 2007, Sriperimbudur & al 2010]*

Example: Compressive K-means

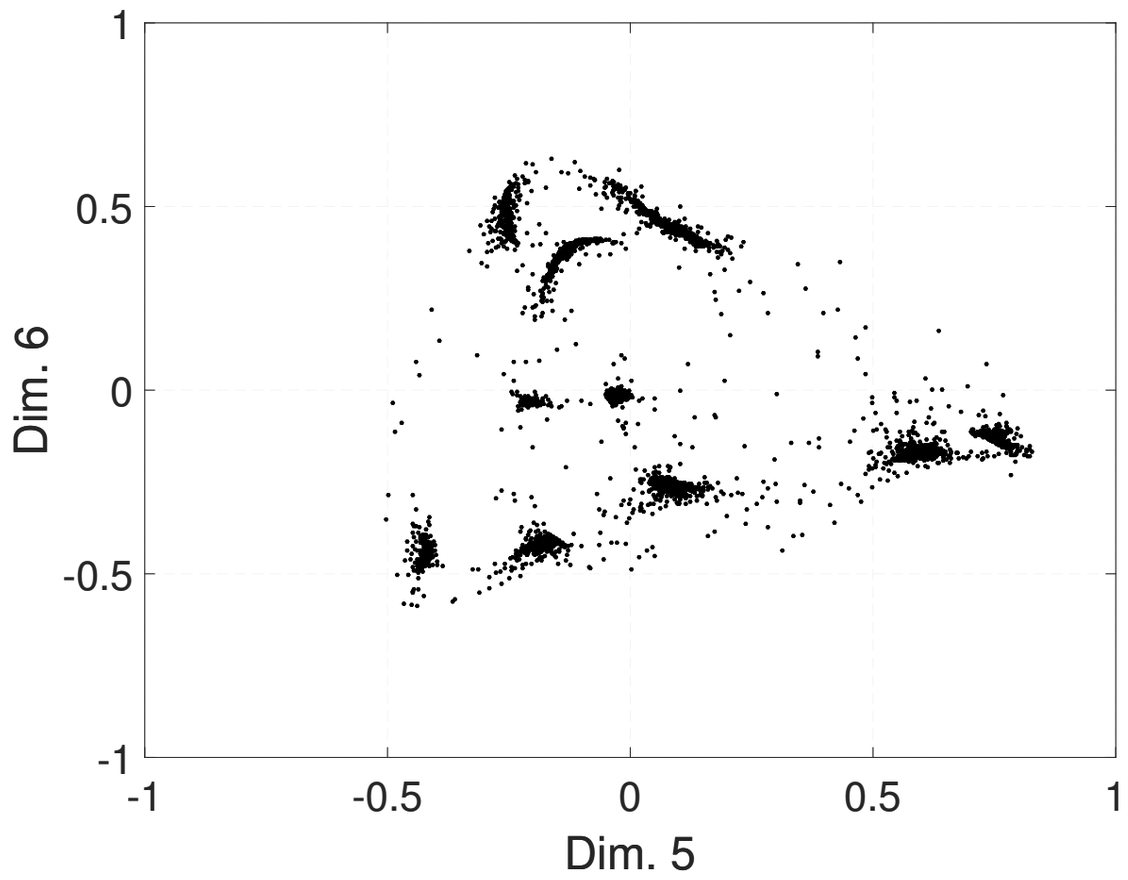
Training set $\mathcal{X}$

$n = 70000;\ d = 784;\ k = 10$

Example: Compressive K-means

Training set $\mathcal{X}$

$n = 70000;\ d = k = 10$

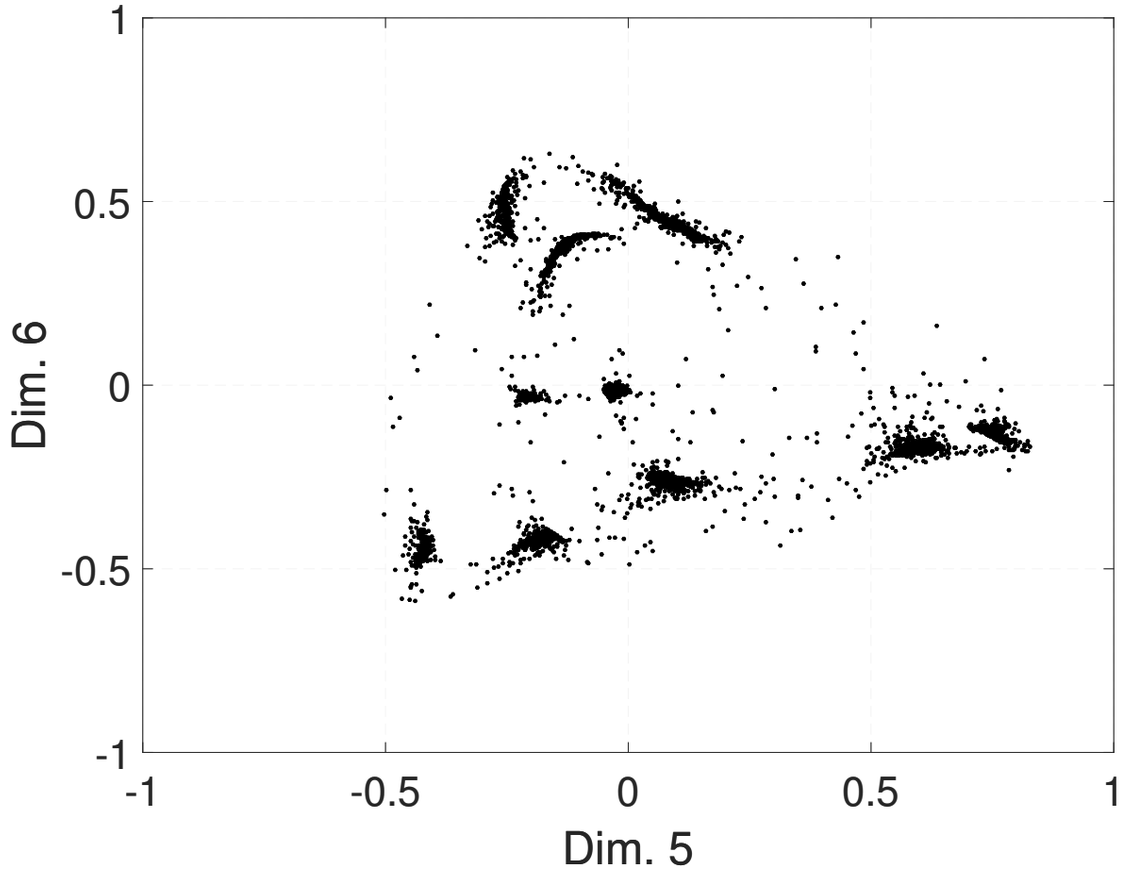


Spectral features

Example: Compressive K-means

Training set $\mathcal{X}$

$$n = 70000;\ d = k = 10$$



Spectral features

$$\mathrm{Sketch}(\mathcal{X})$$

streaming / distributed computation

Sketch vector

memory size *independent* of *n*

$$m \gtrsim kd = 100$$

$$z \in \mathbb{R}^m = \frac{1}{n}\sum_{i=1}^{n} \Phi(x_i)$$

Example: Compressive K-means

$$n = 70000;\ d = k = 10$$

Sketch vector

memory size *independent* of *n*

$$m \gtrsim kd = 100$$

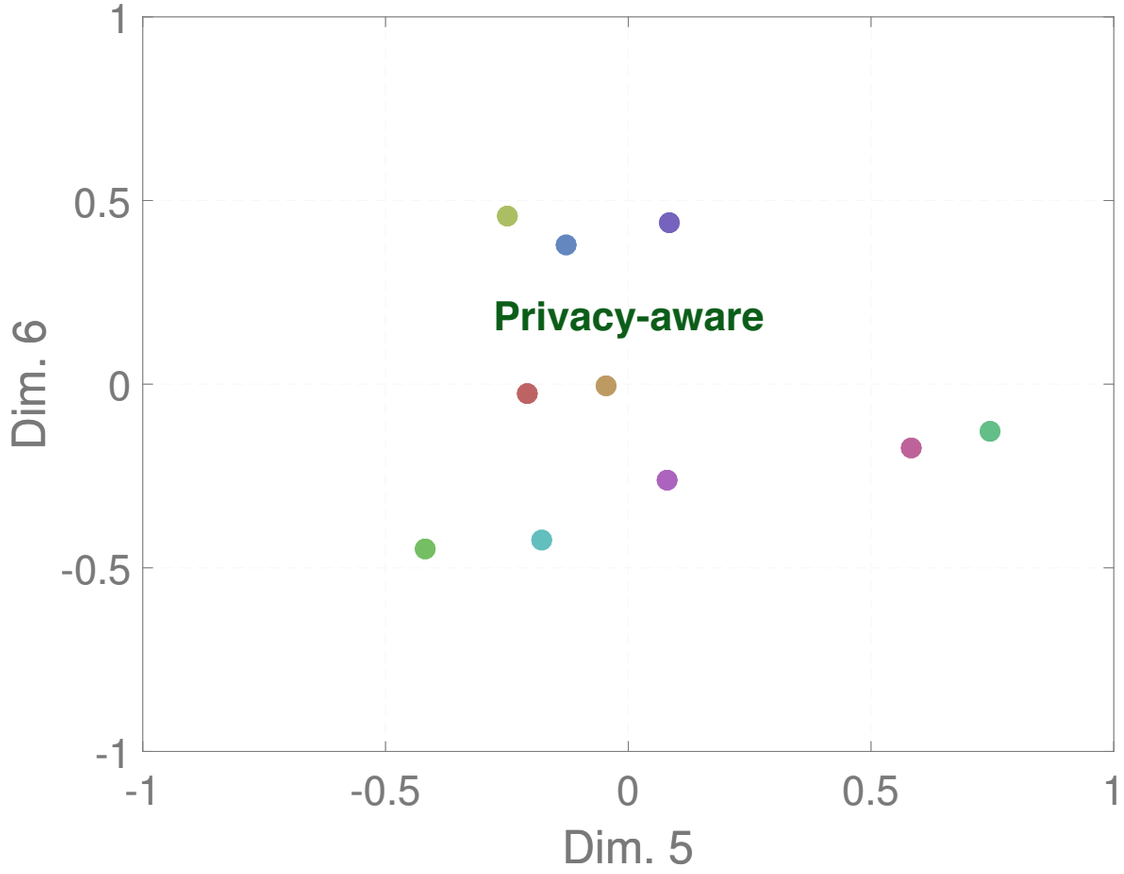
$$z \in \mathbb{R}^m = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)$$

Privacy-aware

streaming / distributed computation

Example: Compressive K-means

$$n = 70000;\ d = k = 10$$



Sketch vector

memory size *independent* of *n*

$$m \gtrsim kd = 100$$

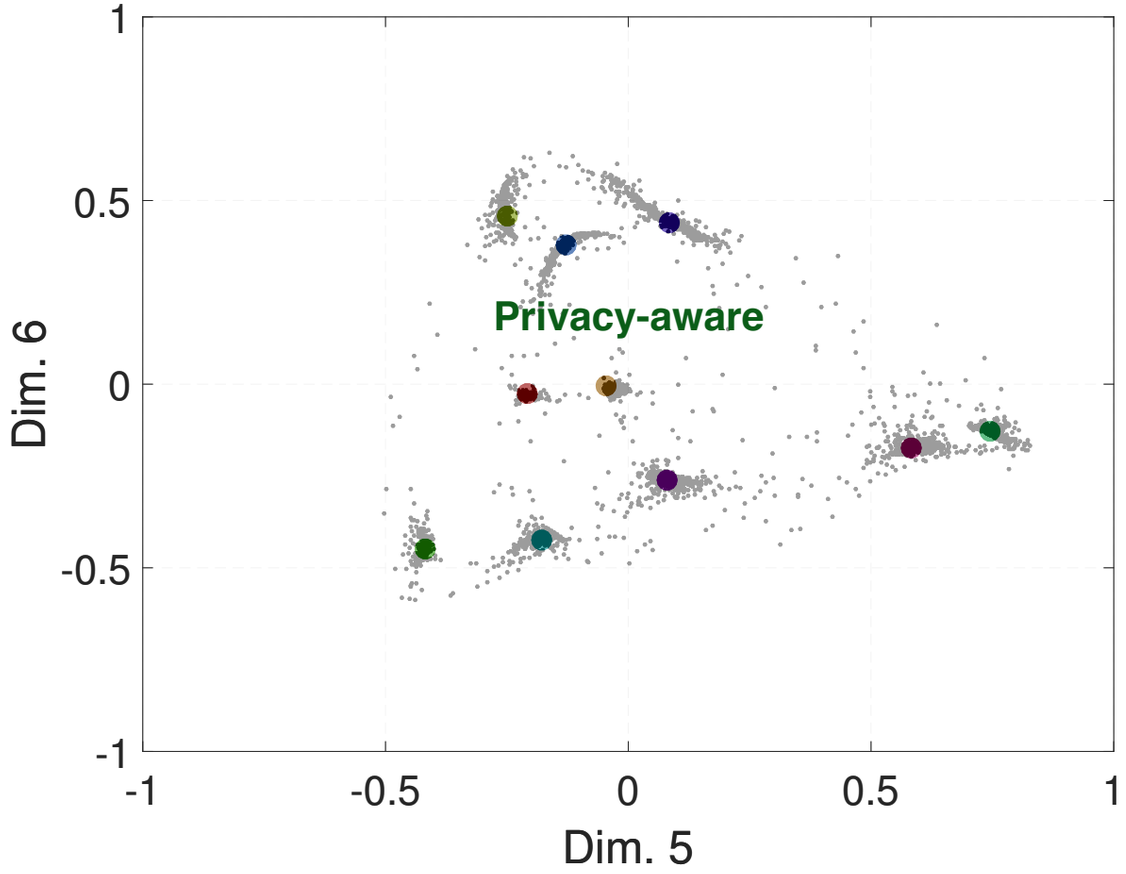
$$z \in \mathbb{R}^m = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)$$

streaming / distributed computation

Learn centroids
from sketch
= moment *fitting*

Example: Compressive K-means

$$n = 70000;\ d = k = 10$$



Sketch vector

memory size *independent* of *n*

$$m \gtrsim kd = 100$$

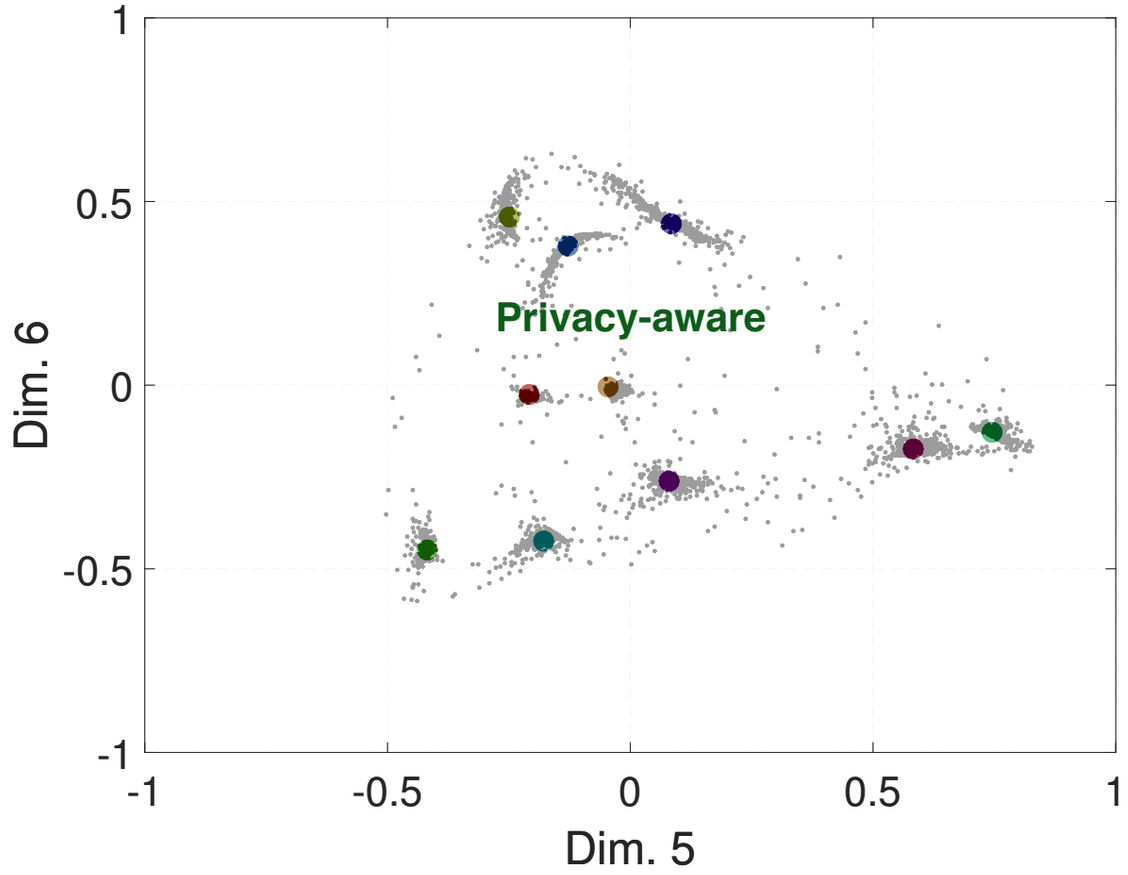
$$z \in \mathbb{R}^m = \frac{1}{n}\sum_{i=1}^{n} \Phi(x_i)$$

streaming / distributed computation

Learn centroids
from sketch
= moment *fitting*

Example: Compressive K-means

$$n = 70000;\ d = k = 10$$



Sketch vector

memory size *independent* of *n*

$$m \gtrsim kd = 100$$

$$z \in \mathbb{R}^m = \frac{1}{n}\sum_{i=1}^{n} \Phi(x_i)$$

streaming / distributed computation

Learn centroids
from sketch
= moment *fitting*

Using: random Fourier features

$$\Phi(x) := \{e^{\imath \omega_j^T x}\}_{j=1}^{m}$$

***Vector*-valued function**

Sketching & Neural networks

- Sketching for k-means
  - empirical characteristic function

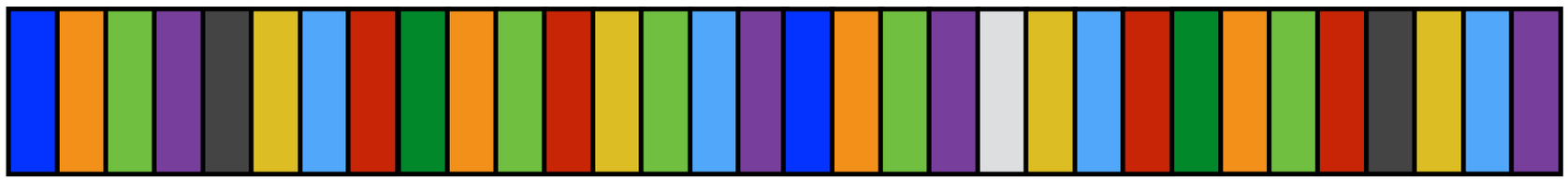
$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$

Sketching & Neural networks

- Sketching for k-means
  - empirical characteristic function

$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$

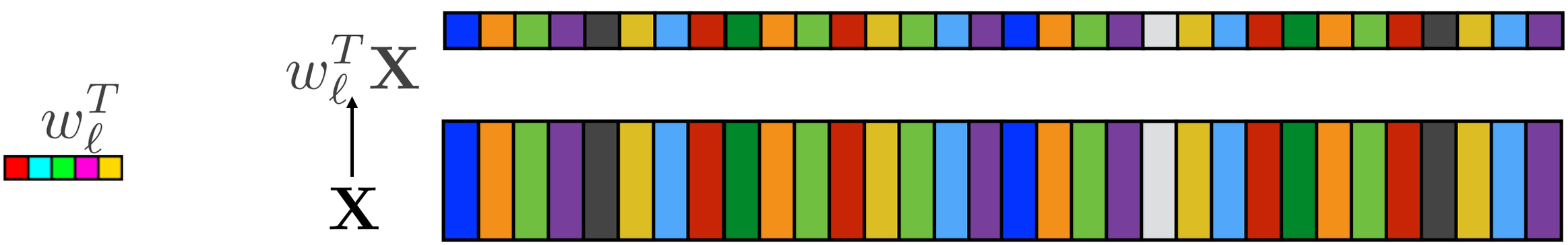
$\mathbf{X}$



Sketching & Neural networks

- Sketching for k-means
  - empirical characteristic function

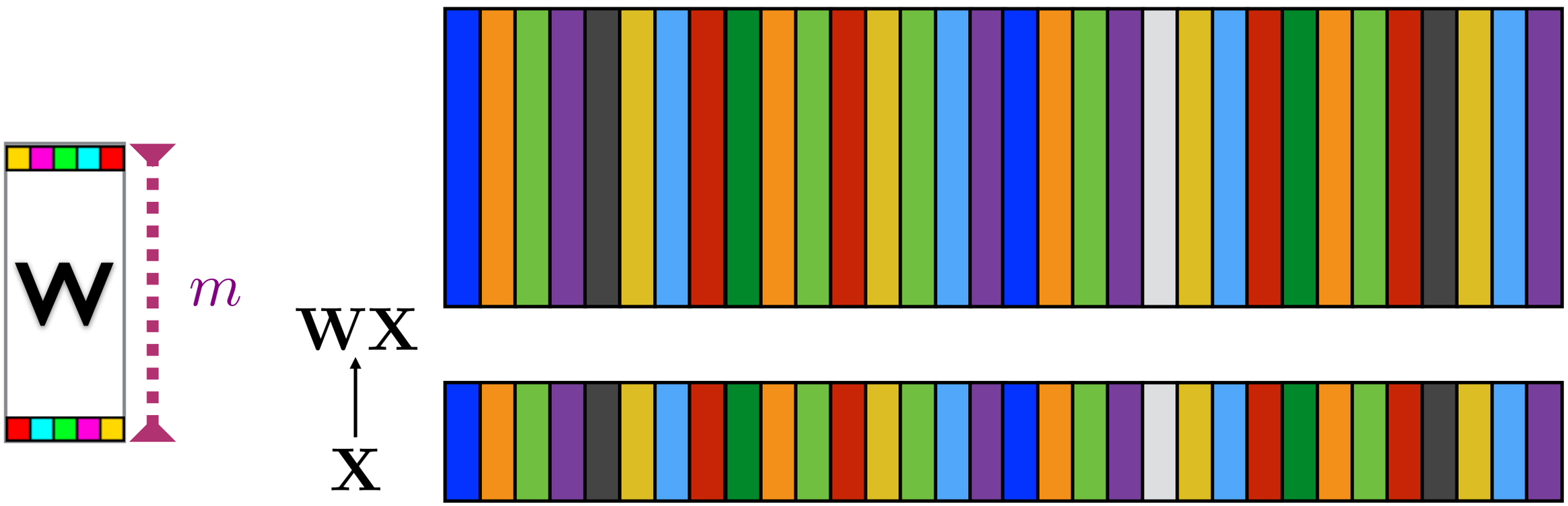
$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$



Sketching & Neural networks

- Sketching for k-means
  - empirical characteristic function

$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$

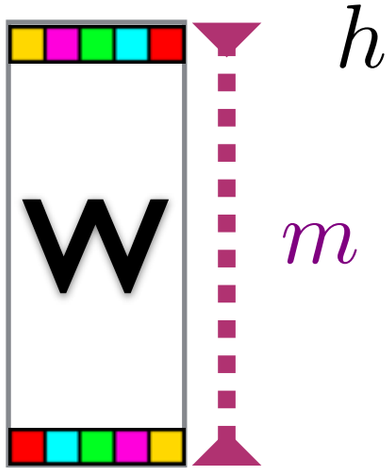


Sketching & Neural networks

- Sketching for k-means
  - empirical characteristic function

$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$

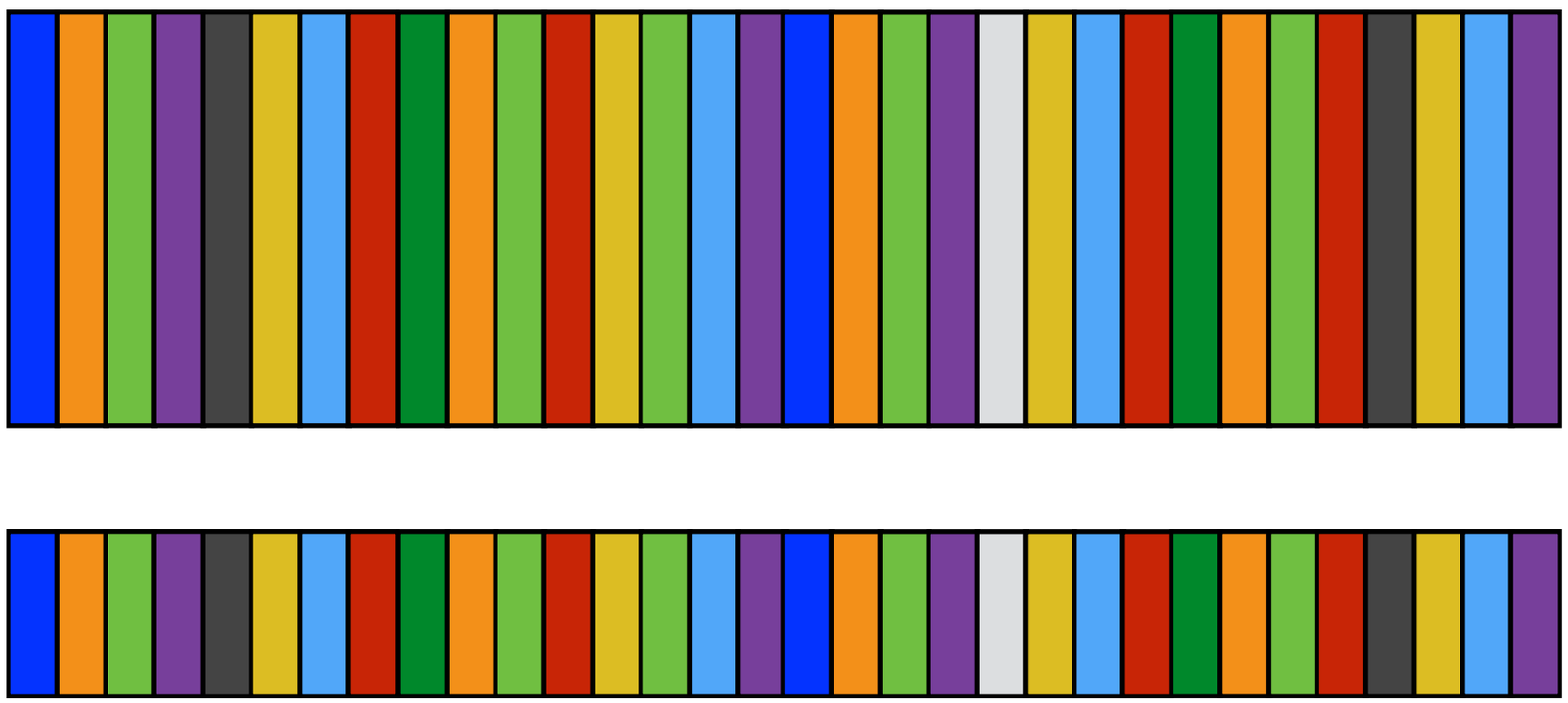
$$h(\cdot) = e^{j(\cdot)}$$



$h(\mathbf{WX})$

$\mathbf{WX}$

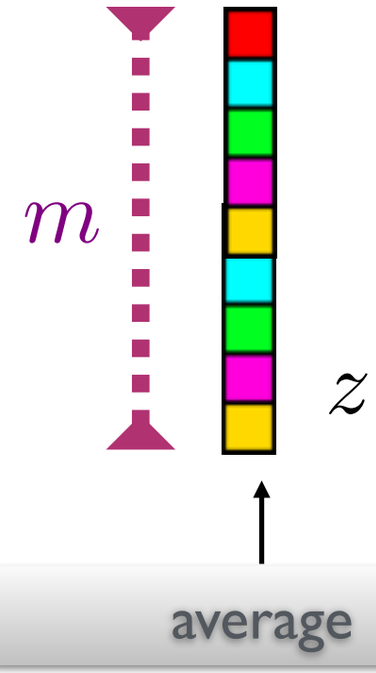
$\mathbf{X}$



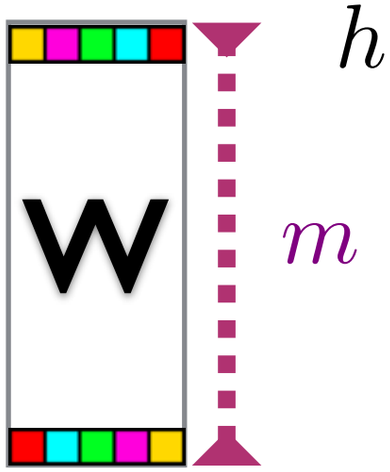
Sketching & Neural networks

- Sketching for k-means
  - empirical characteristic function

$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$



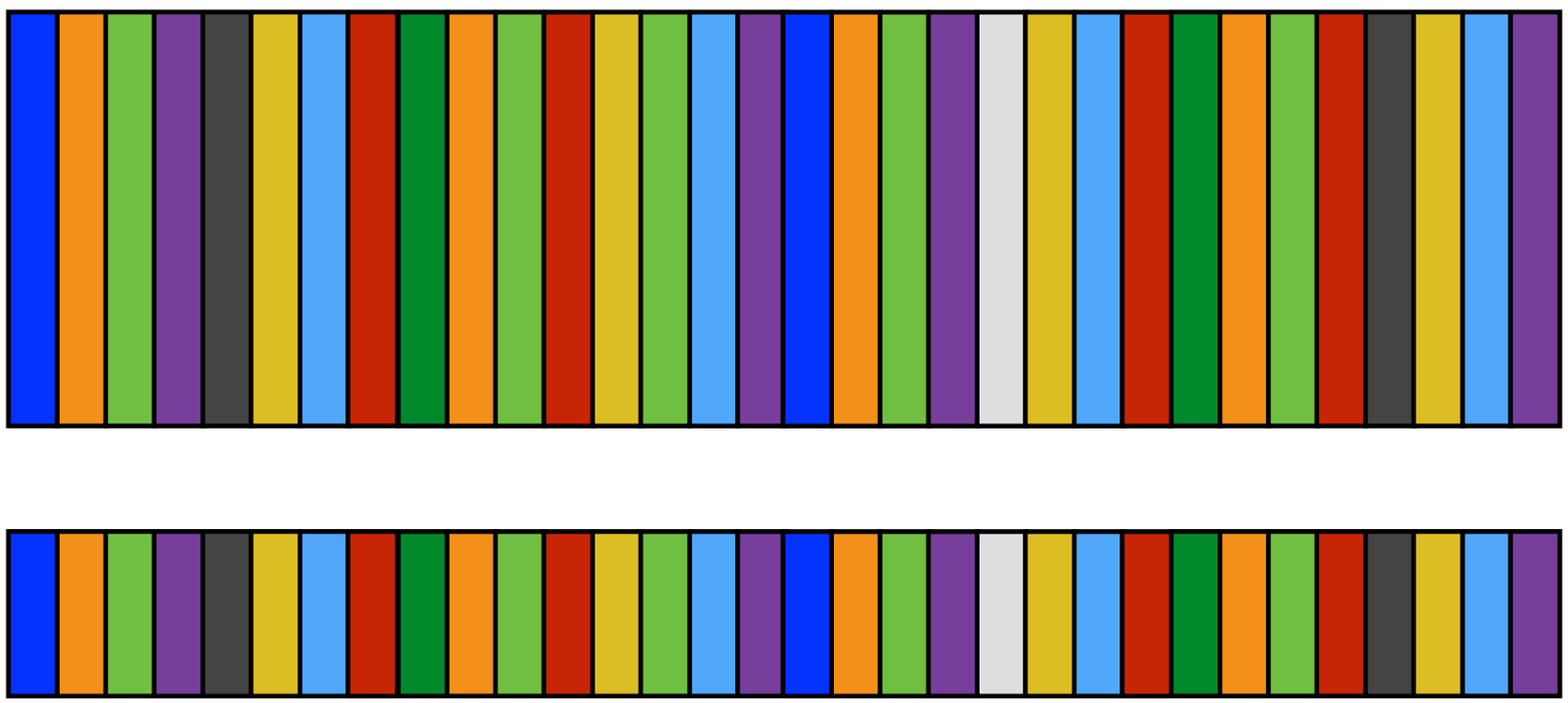
$$h(\cdot) = e^{j(\cdot)}$$



$h(\mathbf{WX})$

$\mathbf{WX}$

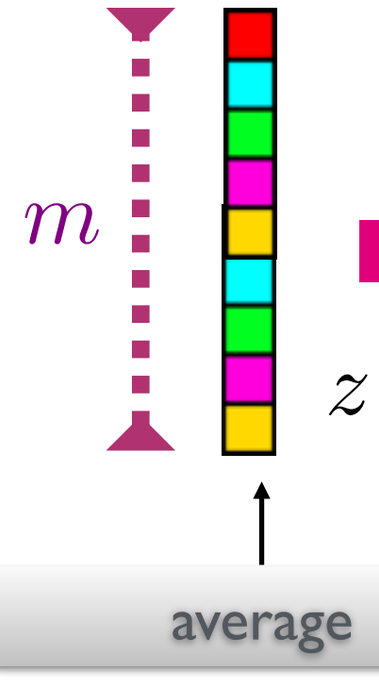
$\mathbf{X}$



Sketching & Neural networks

- Sketching for k-means
  - empirical characteristic function

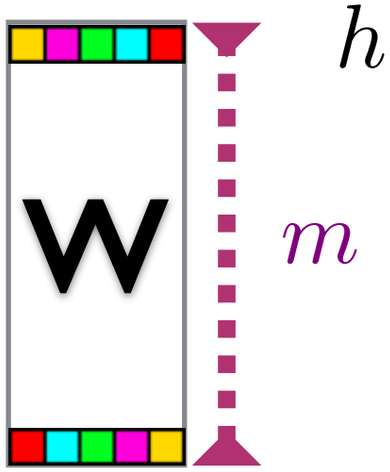
$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$



- ~ One-layer random neural net
  - DNN ~ hierarchical sketching ?

see also [Bruna & al 2013, Giryes & al 2015]

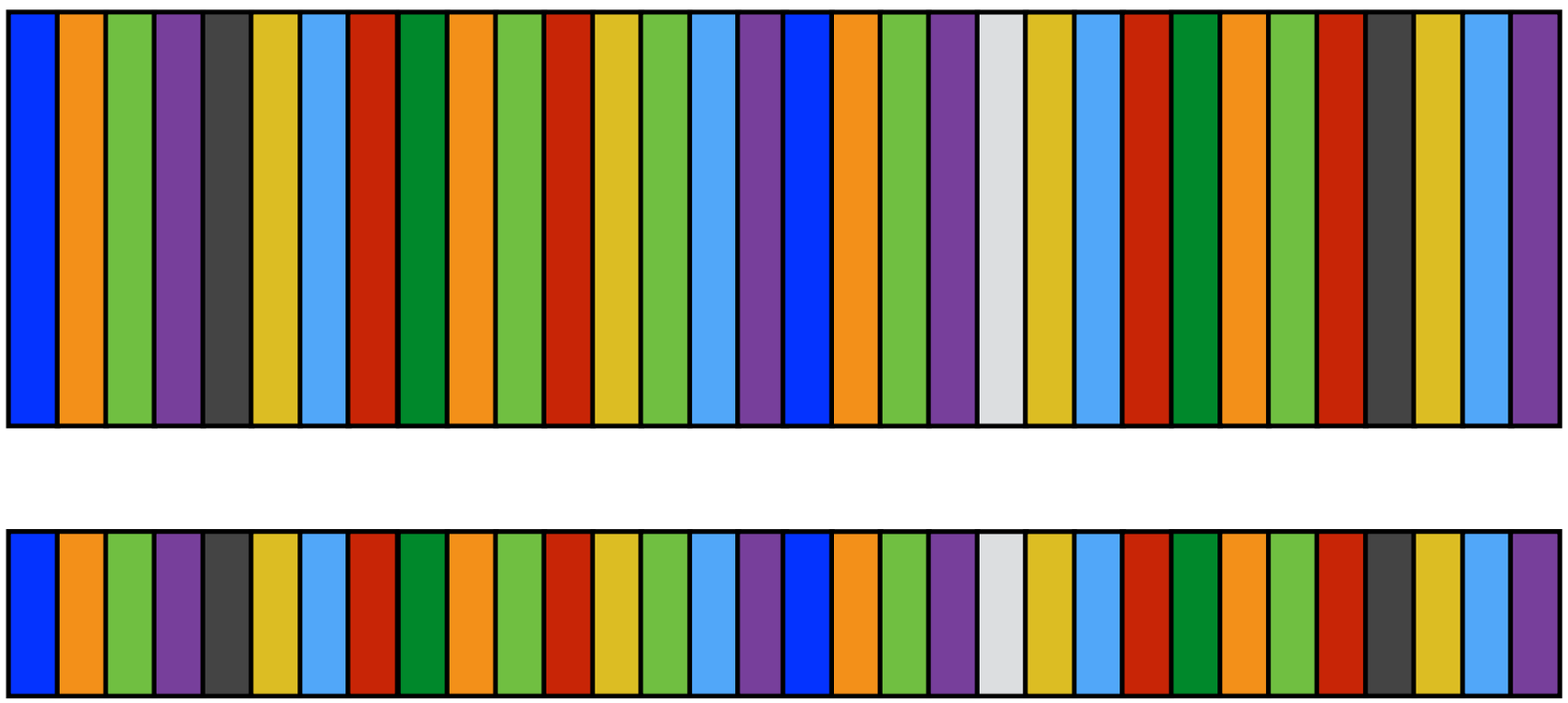
$$h(\cdot) = e^{j(\cdot)}$$



$h(\mathbf{WX})$

$\mathbf{WX}$

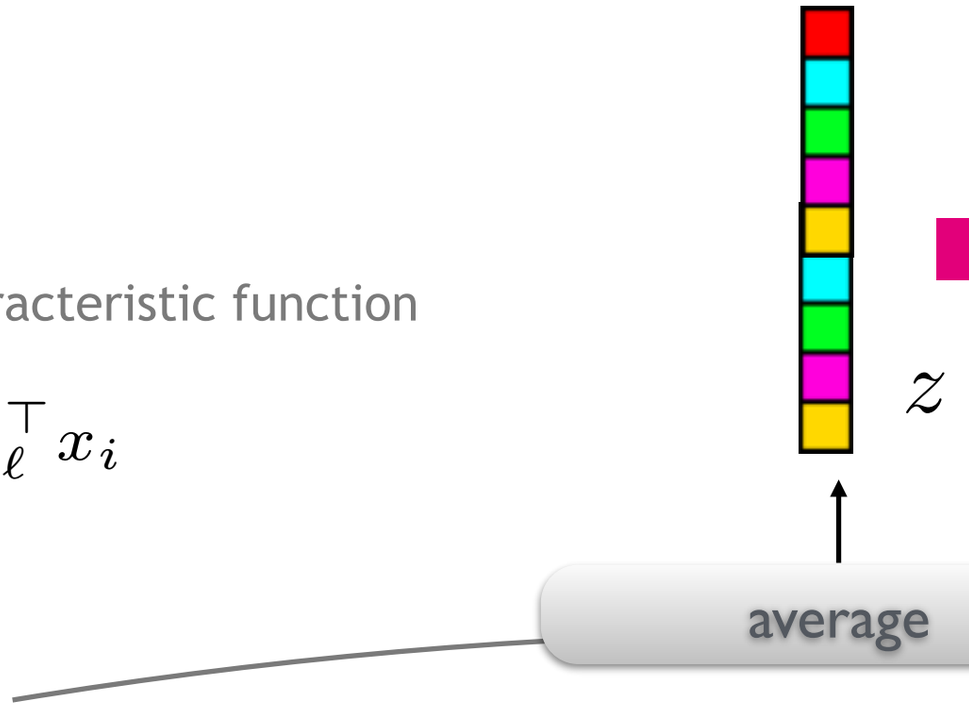
$\mathbf{X}$



Sketching & Privacy

- Sketching
  - empirical characteristic function

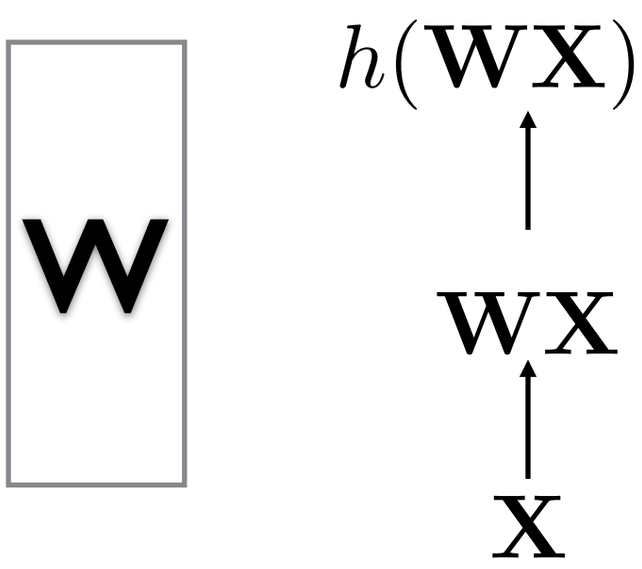
$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$



- ~ One-layer random neural net
  - DNN ~ hierarchical sketching ?

see also [Bruna & al 2013, Giryes & al 2015]

$$h(\cdot) = e^{j(\cdot)}$$



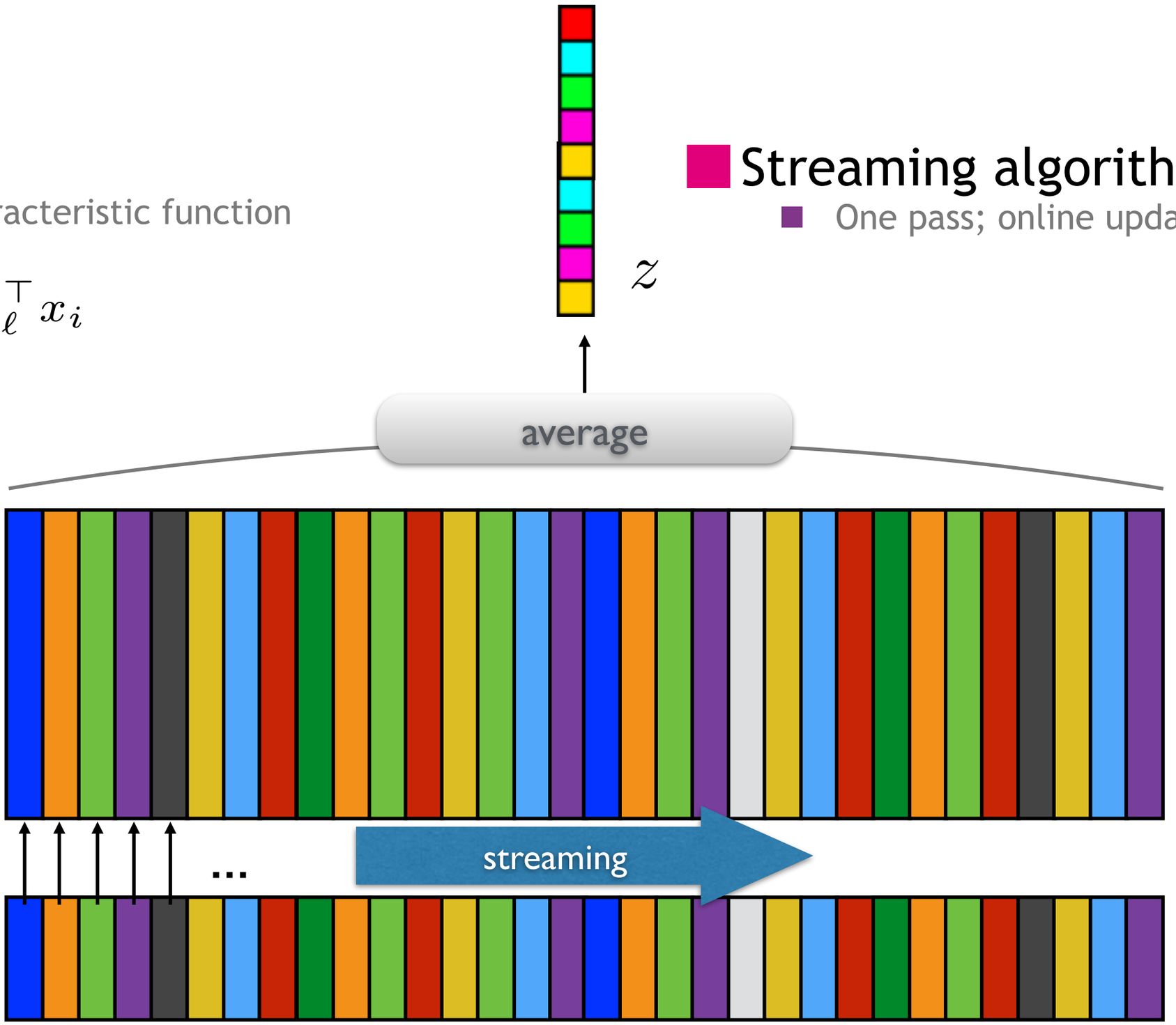
- Privacy-reserving
  - sketch and forget

Sketching & Online Learning

Sketching

- empirical characteristic function

$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$



Streaming algorithms

- One pass; online update

Sketching & Distributed Computing

Sketching

- empirical characteristic function

$$z_\ell = \frac{1}{n} \sum_{i=1}^{n} e^{j w_\ell^\top x_i}$$

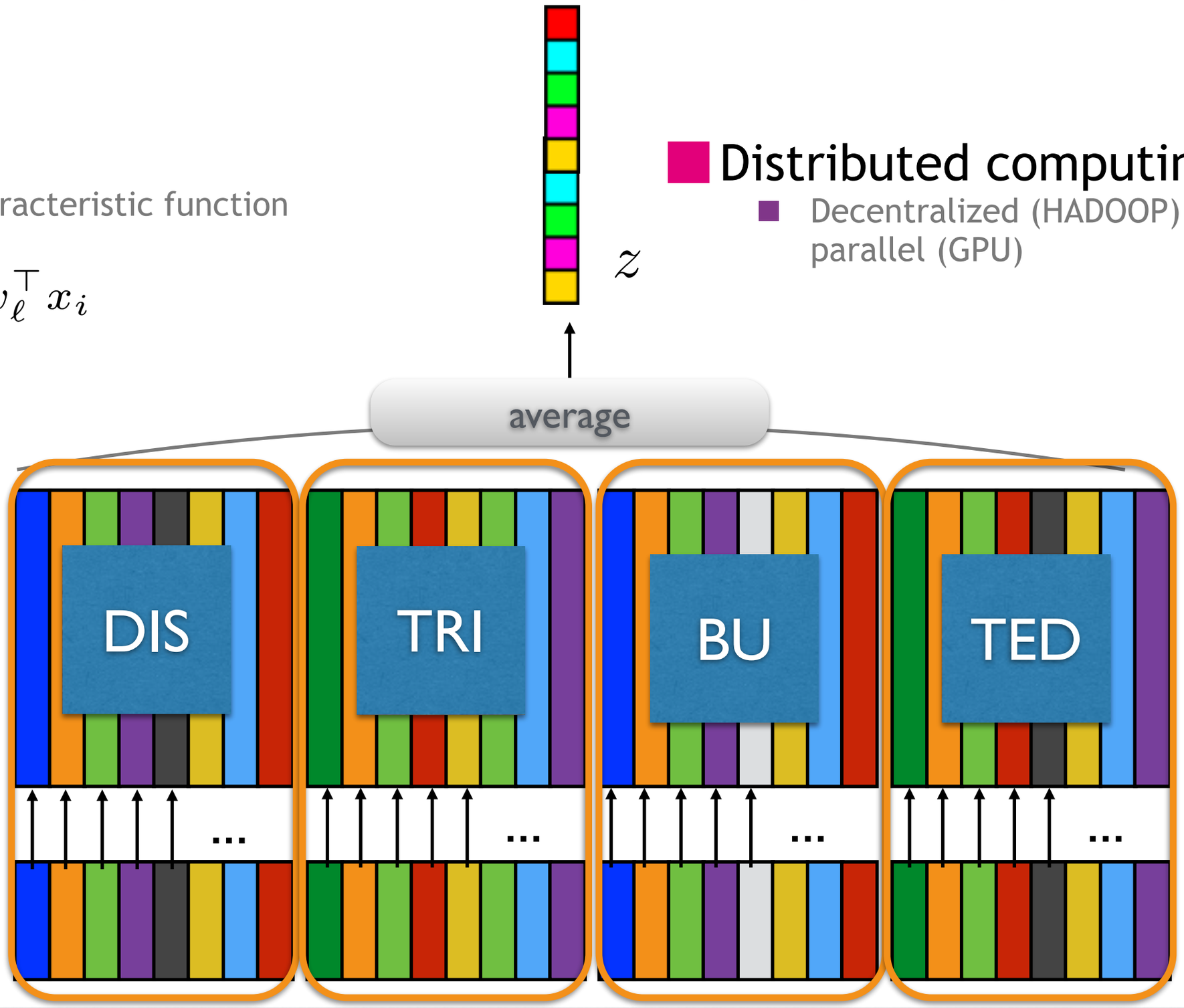
$$h(\cdot) = e^{j(\cdot)}$$

$$h(\mathbf{W}\mathbf{X})$$

$$\mathbf{W}\mathbf{X}$$

$$\mathbf{X}$$

$\mathbf{W}$



Distributed computing

- Decentralized (HADOOP) / parallel (GPU)

- Learning from random moments: the concept
- Compressive *Statistical* Learning (guarantees)
- Recent developments & perspectives

Statistical learning 101

- **Statistical risk** $R(p, \theta) = \mathbb{E}_{x \sim p} \ell(x, \theta)$

- **Target** $\theta^\star \in \arg\min_\theta R(p^\star, \theta)$

- **Empirical version** $\hat{\theta}_n \in \arg\min_\theta R(\hat{p}_n, \theta)$

  $x_i \sim p^\star$, i.i.d. $\quad \hat{p}_n := \frac{1}{n} \sum_{i=1}^n \delta_{x_i}$

- **PAC / *excess risk control / generalization error***

$$R(p^\star, \hat{\theta}_n) \leq R(p^\star, \theta^\star) + \eta_n$$

  - *can be achieved if uniform convergence, i.e. whp* $\sup_\theta |R(\hat{p}_n, \theta) - R(p^\star, \theta)| \leq \eta_n / 2$

Compressive Statistical Learning

Step 1: given learning task

- Design ***sketching function*** $\Phi(x) \in \mathbb{R}^m$

Step 2: compress & learn

- Summarize training collection with sketch

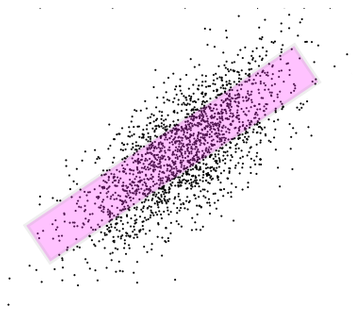
$$z = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)$$

- Learn from sketch with ***some algorithm***

$$z \mapsto \hat{\theta}(z)$$

➡ *controlled excess risk (PAC)?* $R(p^\star, \hat{\theta}(z)) \leq R(p^\star, \theta^\star) + \eta_n$

Worked example 1: Compressive PCA



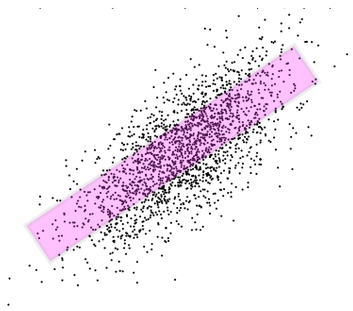
- **Task:** select a *k-dimensional subspace* θ
  - Loss function:

$$\ell(x,\theta) = \|x - P_\theta x\|^2 \qquad x \in \mathbb{R}^d$$

- **Step 1: hoice of sketching function**
  - **naive:** full covariance matrix, $\mathbf{\Sigma} = \mathbb{E} X X^T$

$$m = \mathcal{O}(d^2) \qquad \Phi(x) = xx^T \qquad z \approx \mathrm{vec}(\mathbf{\Sigma})$$

Worked example 1: Compressive PCA



- **Task:** select a *k-dimensional subspace* θ
  - Loss function:

$$\ell(x,\theta) = \|x - P_\theta x\|^2 \qquad x \in \mathbb{R}^d$$

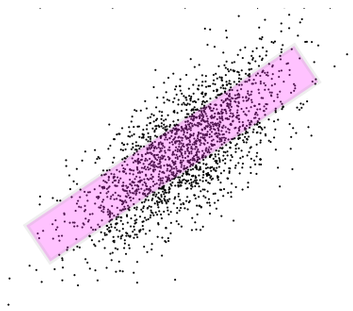
- **Step 1: hoice of sketching function**
  - **naive:** full covariance matrix, $\mathbf{\Sigma} = \mathbb{E} X X^T$

$$m = \mathcal{O}(d^2) \qquad \Phi(x) = xx^T \qquad z \approx \mathrm{vec}(\mathbf{\Sigma})$$

  - **refined:**= using compressive matrix sensing

$$m = \mathcal{O}(kd) \qquad \Phi(x) = \{(\omega_j^T x)^2\}_{j=1}^m \qquad z \approx A(\mathrm{vec}(\mathbf{\Sigma}))$$

Worked example 1: Compressive PCA



- **Task:** select a *k-dimensional subspace* θ
  - Loss function:

$$\ell(x,\theta) = \|x - P_\theta x\|^2 \qquad x \in \mathbb{R}^d$$

- **Step 1: hoice of sketching function**
  - **naive:** full covariance matrix, $\mathbf{\Sigma} = \mathbb{E} X X^T$

$$m = \mathcal{O}(d^2) \qquad \Phi(x) = xx^T \qquad z \approx \mathrm{vec}(\mathbf{\Sigma})$$

  - **refined:**= using compressive matrix sensing

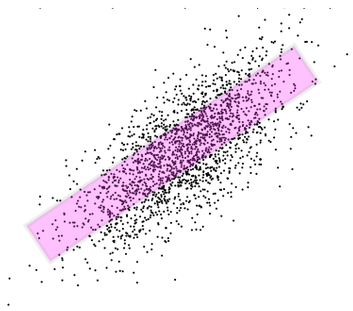
$$m = \mathcal{O}(kd) \qquad \Phi(x) = \{(\omega_j^T x)^2\}_{j=1}^m \qquad z \approx A(\mathrm{vec}(\mathbf{\Sigma}))$$

- **Learn from sketch:** low-rank matrix recovery

$$\hat{\theta}(z) := \mathrm{span}(\hat{\mathbf{\Sigma}}_k(z)) \qquad \hat{\mathbf{\Sigma}}_k(z) := \arg \min_{\substack{\mathrm{rank}\mathbf{\Sigma} \leq k \\ \mathbf{\Sigma} \succeq 0}} \|z - A(\mathrm{vec}(\mathbf{\Sigma}))\|_2$$

✓ *statistical guarantees* $\quad R(\hat{\theta}) \leq (1+C) R(\theta^*) + O(1/\sqrt{n})$

Worked example 1: Compressive PCA



- **Task:** select a *k-dimensional subspace* θ
  - Loss function:

$$\ell(x,\theta) = \|x - P_\theta x\|^2 \qquad x \in \mathbb{R}^d$$

- **Step 1: hoice of sketching function**
  - **naive:** full covariance matrix, $\mathbf{\Sigma} = \mathbb{E} X X^T$

$$m = \mathcal{O}(d^2) \qquad \Phi(x) = xx^T \qquad z \approx \mathrm{vec}(\mathbf{\Sigma})$$

  - **refined:**= using compressive matrix sensing

$$m = \mathcal{O}(kd)$$

✓ # of parameters to learn

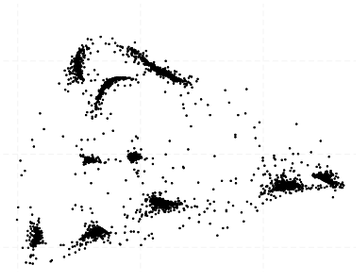
$$\Phi(x) = \{(\omega_j^T x)^2\}_{j=1}^m \qquad z \approx A(\mathrm{vec}(\mathbf{\Sigma}))$$

- **Learn from sketch:** low-rank matrix recovery

$$\hat{\theta}(z) := \mathrm{span}(\hat{\mathbf{\Sigma}}_k(z)) \qquad \hat{\mathbf{\Sigma}}_k(z) := \arg \min_{\substack{\mathrm{rank}\mathbf{\Sigma} \leq k \\ \mathbf{\Sigma} \succeq 0}} \|z - A(\mathrm{vec}(\mathbf{\Sigma}))\|_2$$

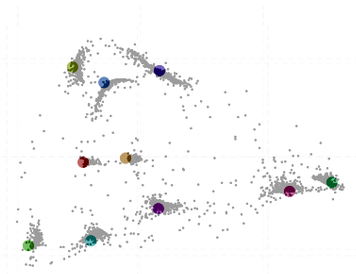
✓ *statistical guarantees* $\quad R(\hat{\theta}) \leq (1+C) R(\theta^*) + O(1/\sqrt{n})$

Worked example 2: Compressive K-Means



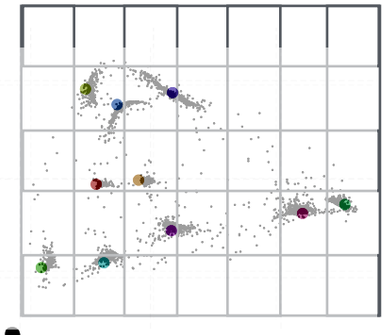
- **Task: find *k* centroids** $\theta = \{\theta_1, \ldots, \theta_k\},\ \theta_i \in \mathbb{R}^d$
  - Loss function: $\ell(x, \theta) = \min_i \|x - \theta_i\|^2$
- **Standard approach: “K-means algorithm”**
  - *aka* Lloyd-Max algorithm [Steinhaus 1956, Lloyd 1957 (publ. 1982)]
    - several passes on the training set

Worked example 2: Compressive K-Means



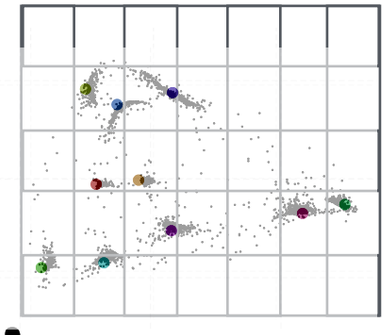
- **Task: find *k* centroids** $\theta = \{\theta_1, \ldots, \theta_k\},\ \theta_i \in \mathbb{R}^d$
  - Loss function: $\ell(x, \theta) = \min_i \|x - \theta_i\|^2$
- **Standard approach: “K-means algorithm”**
  - *aka* Lloyd-Max algorithm [Steinhaus 1956, Lloyd 1957 (publ. 1982)]
    - several passes on the training set

Worked example 2: Compressive K-Means



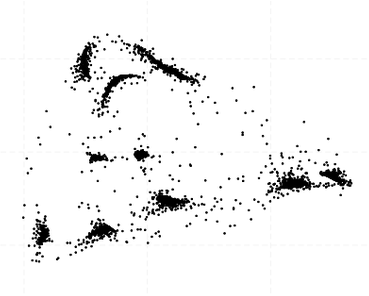
- **Task: find *k* centroids** $\theta = \{\theta_1, \ldots, \theta_k\},\ \theta_i \in \mathbb{R}^d$
  - Loss function: $\ell(x, \theta) = \min_i \|x - \theta_i\|^2$
- **Standard approach: "K-means algorithm"**
  - *aka* Lloyd-Max algorithm [Steinhaus 1956, Lloyd 1957 (publ. 1982)]
    - several passes on the training set
- **Naive sketching =histograms**
  - $N = \mathcal{O}((R/\epsilon)^d)$ bins of size ϵ within domain of radius *R*
    - exponential sketch size $m = N$

Worked example 2: Compressive K-Means



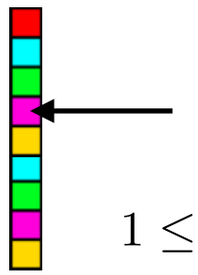
- **Task: find *k* centroids** $\theta = \{\theta_1, \ldots, \theta_k\},\ \theta_i \in \mathbb{R}^d$
  - Loss function: $\ell(x, \theta) = \min_i \|x - \theta_i\|^2$
- **Standard approach: "K-means algorithm"**
  - *aka* Lloyd-Max algorithm [Steinhaus 1956, Lloyd 1957 (publ. 1982)]
    - several passes on the training set
- **Naive sketching =histograms**
  - $N = \mathcal{O}((R/\epsilon)^d)$ bins of size ϵ within domain of radius *R*
    - exponential sketch size $m = N$
    - compressive sensing suggests $m = \mathcal{O}(k \log N) = \mathcal{O}(kd \log(R/\epsilon))$
    - ***can we avoid discretization (bypass curse of dimensionality) ?***

Compressive K-means: How to Sketch ?



Choice of the sketching function ?

- **Observation**: distribution *p(x)* is ***spatially localized***
- **Intuition (from compressive sensing)**
  - need "**incoherent**" sampling
  - choose **Fourier** measurements
  - = ***empirical characteristic function***



$$z_\ell \approx \mathbb{E}_{X\sim p} e^{j w_\ell^\top X} \qquad \omega_\ell \in \mathbb{R}^d$$

$$1 \le \ell \le m$$

- *Sketching function* $\Phi(x) = \frac{1}{\sqrt{m}} \left(e^{j\omega_\ell^\top x} \right)_{\ell=1}^{m}$ **Random Fourier Features** *[Rahimi & Recht 2007]*
- *Sketch vector* z = **Random Fourier *Moments***

Learning centroids from a sketch ?

- **Learning *principle* = moment fitting**
  - Parametric optimization problem

$$\hat{\theta}(z) = \arg \min_{\theta_j \in \mathbb{R}^d} \min_{\alpha_j} \|z - \sum_{j=1}^{k} \alpha_j \Phi(\theta_j)\|_2$$

✓Statistical guarantees (assume ϵ-*separated* centroids)

$$m \approx \mathcal{O}(k^2 d \log(R/\epsilon))$$

compare FoCM 2017 version: $m \approx \mathcal{O}(k^2 d^2 \log(R/\epsilon))$

Learning centroids from a sketch ?

- **Learning *principle* = moment fitting**
  - Parametric optimization problem

$$\hat{\theta}(z) = \arg \min_{\theta_j \in \mathbb{R}^d} \min_{\alpha_j} \|z - \sum_{j=1}^{k} \alpha_j \Phi(\theta_j)\|_2$$

✓Statistical guarantees (assume ϵ-*separated* centroids)

$$m \approx \mathcal{O}(k^2 d \log(R/\epsilon))$$

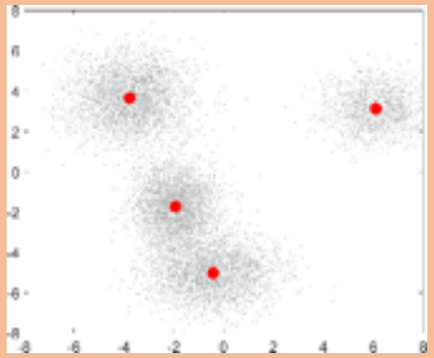
compare FoCM 2017 version: $m \approx \mathcal{O}(k^2 d^2 \log(R/\epsilon))$

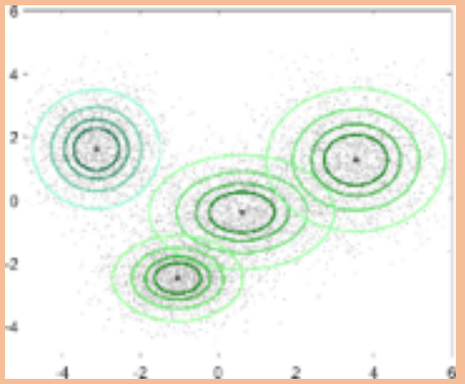
- **Empirical learning *algorithms***
  - Inspiration: sparse recovery algorithms
    - Discretization + convex relaxation [Bunea & al 2010]
    - Convex optimization over (sparse) Radon measures [e.g. Bredies & al 2013]
    - CL-OMPR: greedy and gridless [Keriven, Bourrier, G. & Perez 2016]
      - MP (Mallat & Zhang 93) > OMP (Pati & al 93) > OMPR (Jain 2011) > **CL-OMPR**
      - **similar to Frank-Wolfe [Bredies & al 2013]**
    - CL-AMP: hybrid approximate message passing [Byrne, G. & Schniter 2017]

CL-OMPR & the SketchMLbox

SketchMLbox (`sketchml.gforge.inria.fr`)

- Mixture of Diracs (« K-means »)
- GMMs with known covariance





- GMMs with unknown diagonal covariance
- **Soon**:
  - Mixtures of alpha-stable
  - Gaussian Locally Linear Mapping *[Deleforge 2014]*

- **Handles generic mixtures** $p = \sum_{j=1}^{k} \alpha_j p_{\theta_j}$

  with user-defined $\mathcal{M}p_\theta, \nabla_\theta \mathcal{M} p_\theta$

Sketch size: theory *vs* experiments

In theory, sufficient to have

$$m \gtrsim \mathcal{O}(k^2 d)$$

Empirically ?

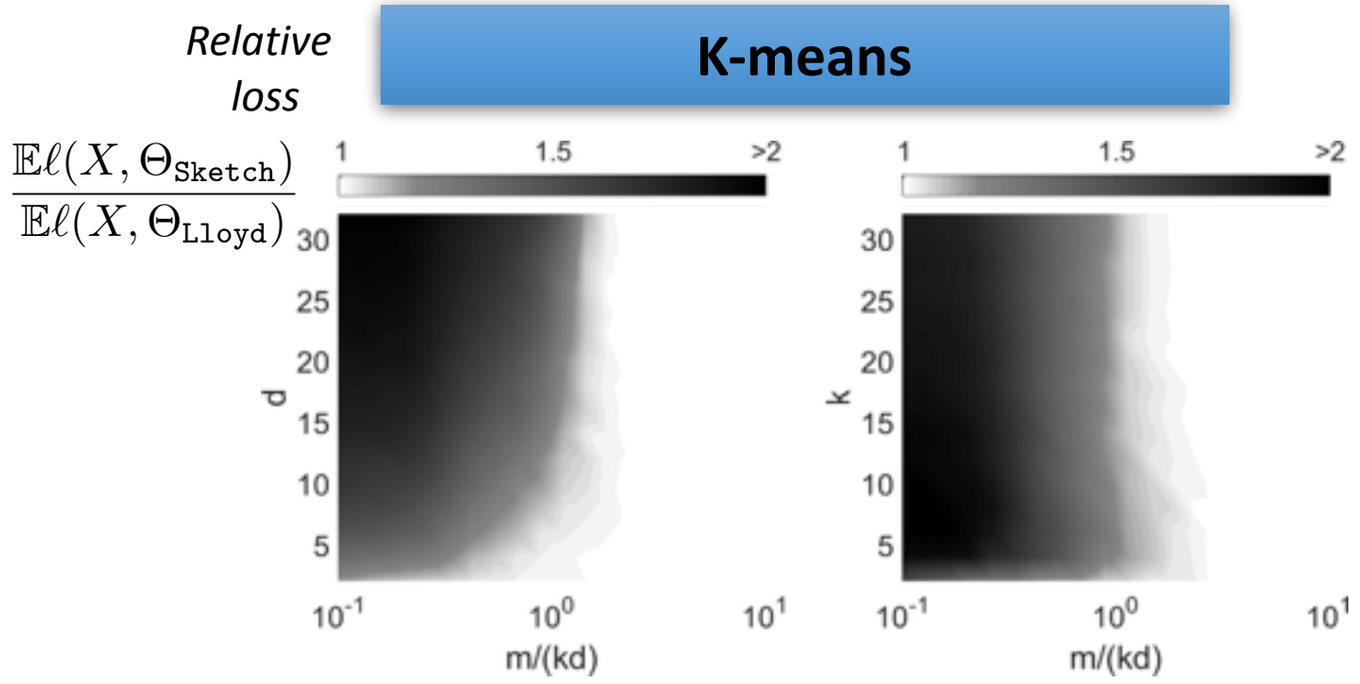
Sketch size: theory *vs* experiments

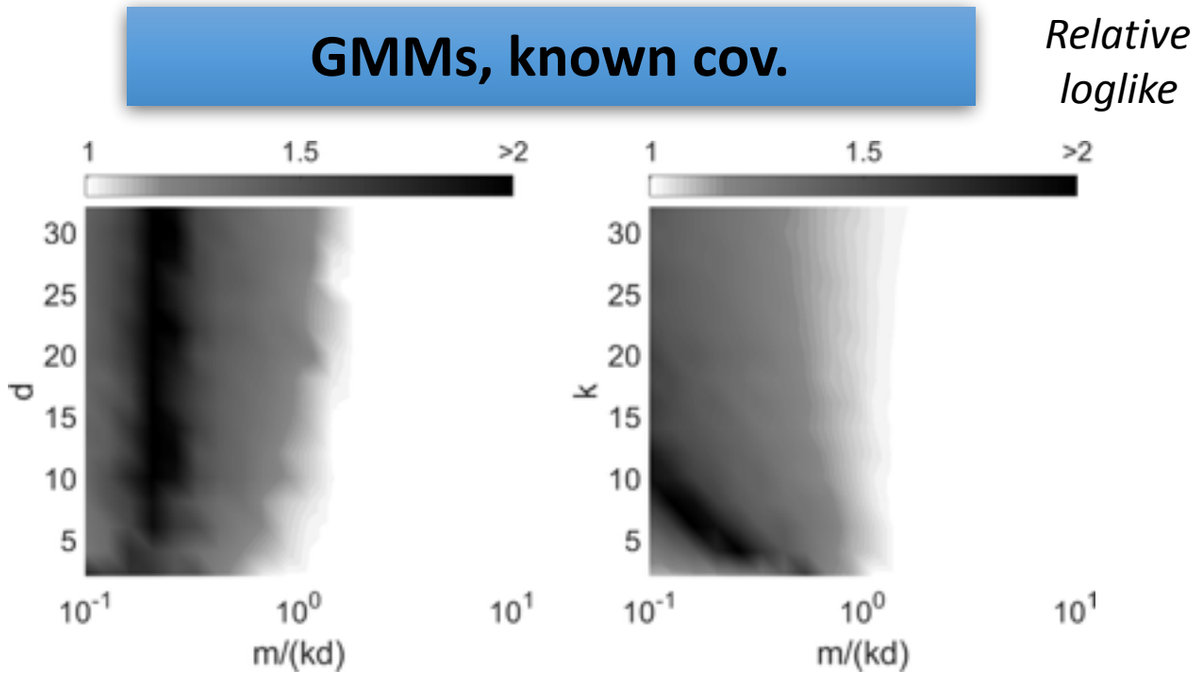
In theory, sufficient to have

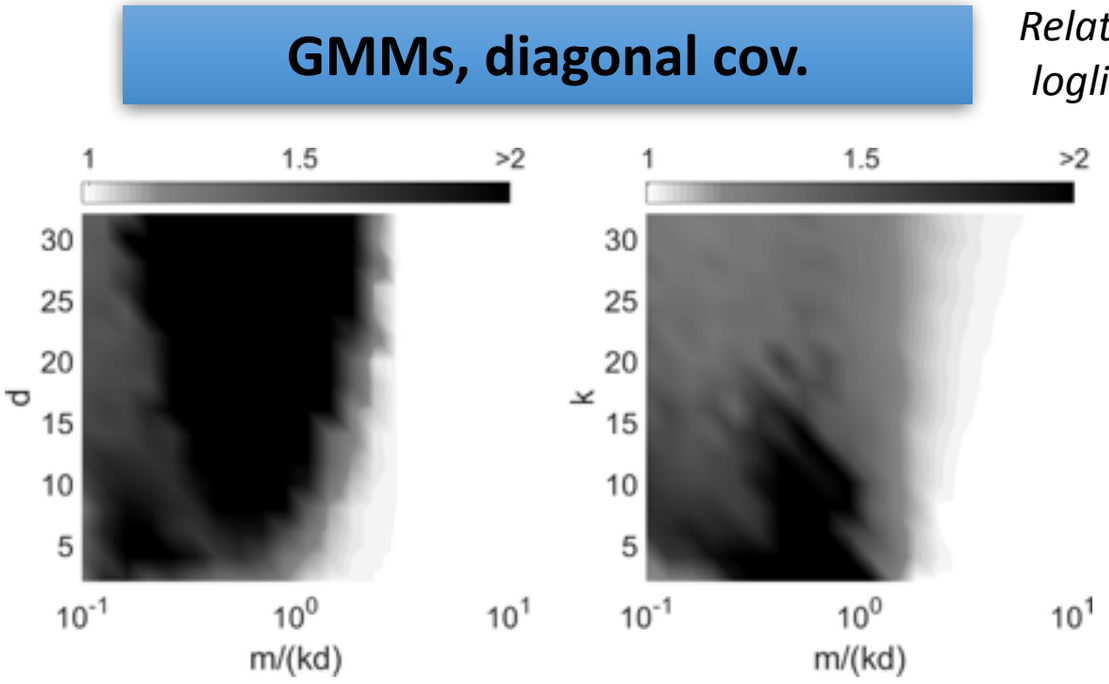
$$m \gtrsim \mathcal{O}(k^2 d)$$

Empirically ?

$$m \approx \mathcal{O}(kd)$$







- Learning from random moments: the concept
- Compressive *Statistical* Learning (guarantees)
- Recent developments & perspectives

Private sketched learning ?

- **“Natural” privacy of an aggregated estimator:**

$$z = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)$$

  - role of sketch size
    - sufficiently large for “task-level” information-preservation
    - *sufficiently small for “sample-level” information loss?*

- **Towards guaranteed differential privacy ?**
  - randomized sketching function ?
    - noise on training samples
    - noise on random features
    - partial random features
    - combinations of the above …

$$\Psi(x_i) = \Phi(x_i + \xi_i)$$
$$\Psi(x_i) = \Phi(x_i) + \xi_i$$
$$\Psi(x_i) = \mathtt{diag}(\mathbf{d}_i) \cdot \Phi(x_i) \in \mathbb{C}^m$$
$$\|\mathbf{d}_i\|_0 = \alpha m, \quad \alpha < 1$$

Hot from the oven

- Compressive *k*-means in $\mathbb{R}^d$

$$k = d = 10$$

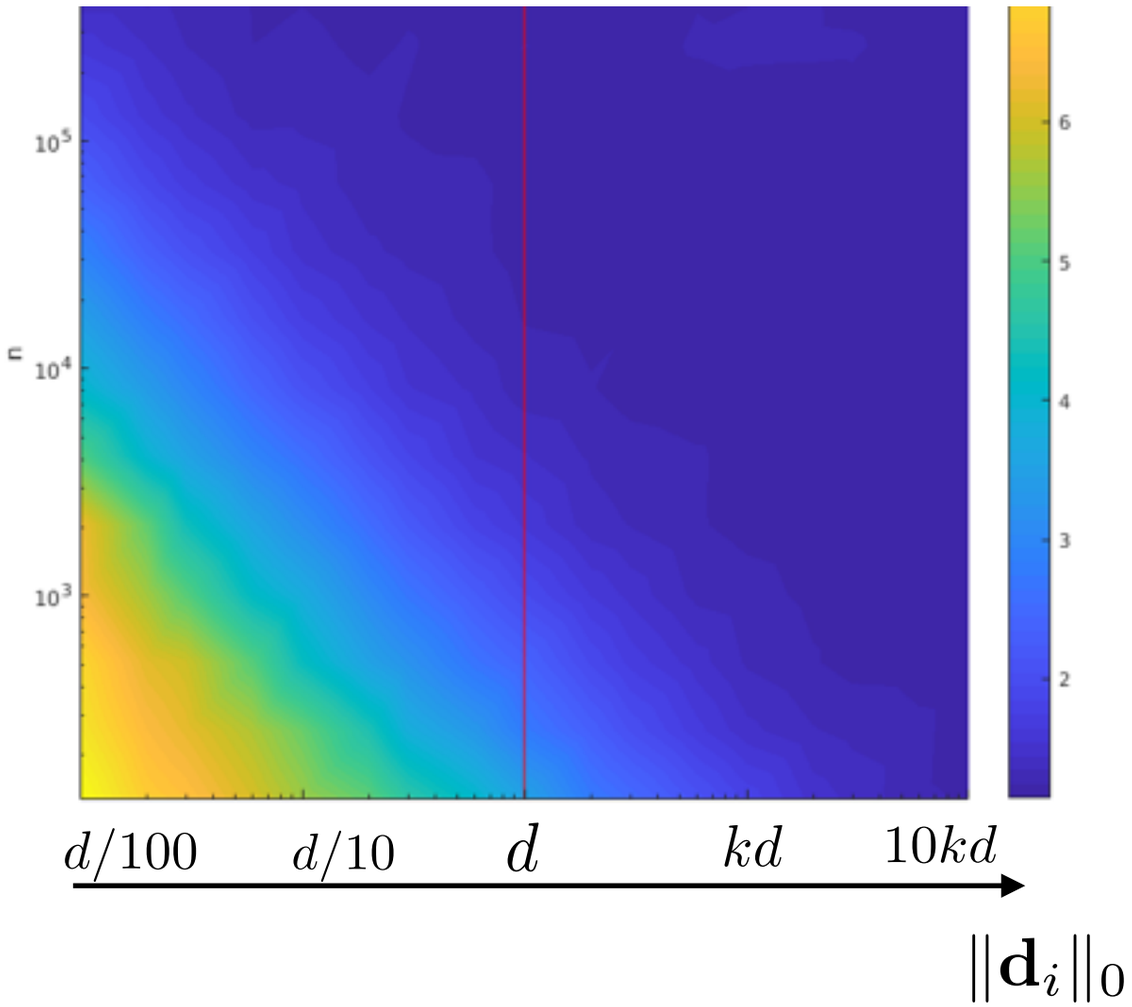
- Sketch size $m = 10kd$

- Private sketching

$$\Psi(x_i) = \mathtt{diag}(\mathbf{d}_i) \cdot \Phi(x_i) \in \mathbb{C}^m$$

  - independent draws of $\mathbf{d}_i$ for each training sample

- Tradeoff privacy / size of training set / quality



Hot from the oven

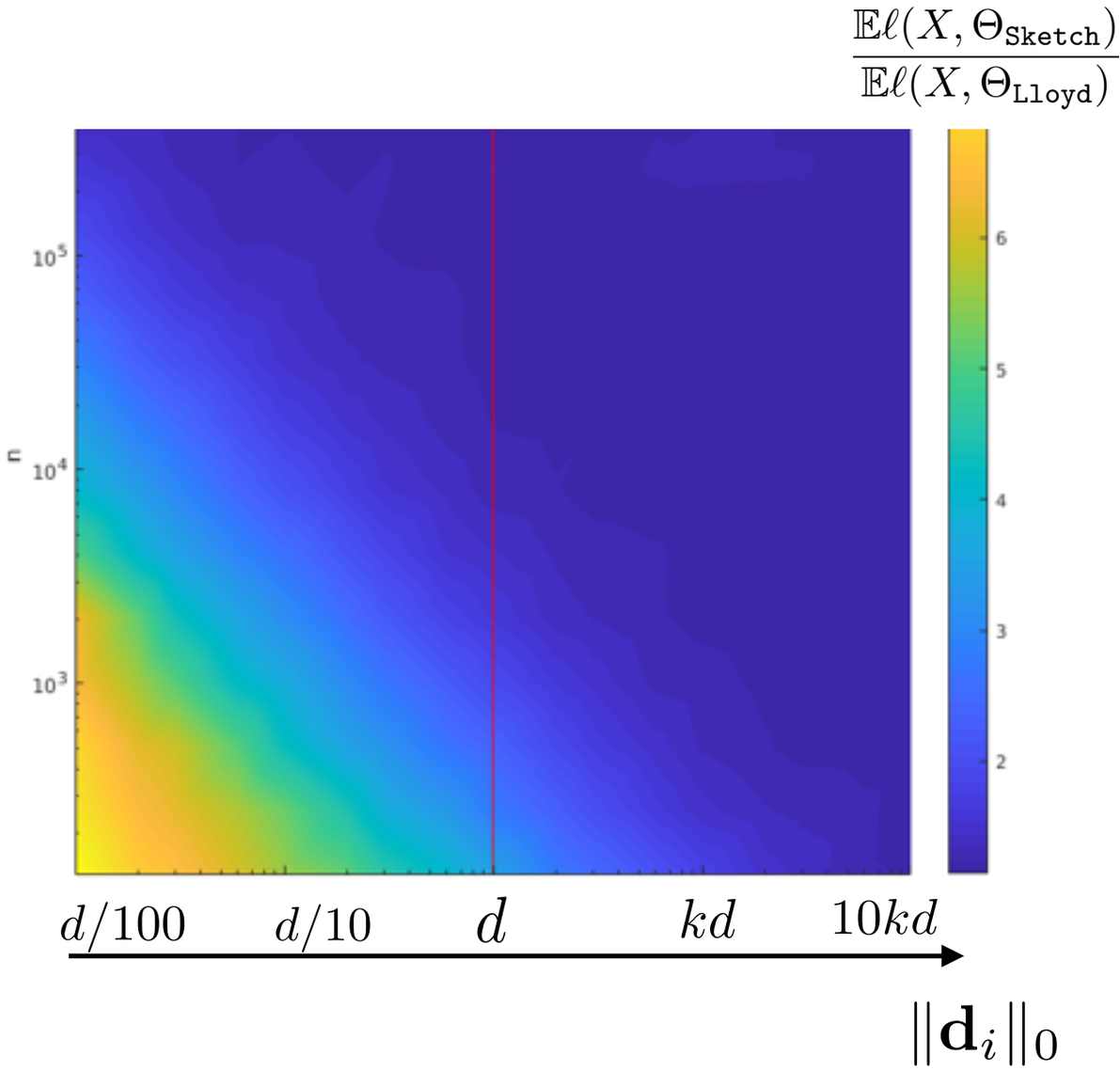
- Compressive *k*-means in $\mathbb{R}^d$
  $k = d = 10$
- Sketch size $m = 10kd$

- Private sketching

$$\Psi(x_i) = \mathtt{diag}(\mathbf{d}_i) \cdot \Phi(x_i) \in \mathbb{C}^m$$

  - independent draws of $\mathbf{d}_i$ for each training sample

- Tradeoff privacy / size of training set / quality

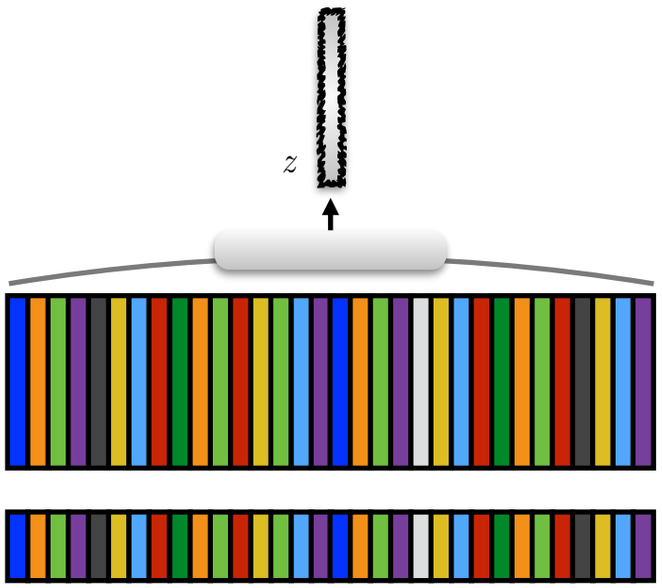


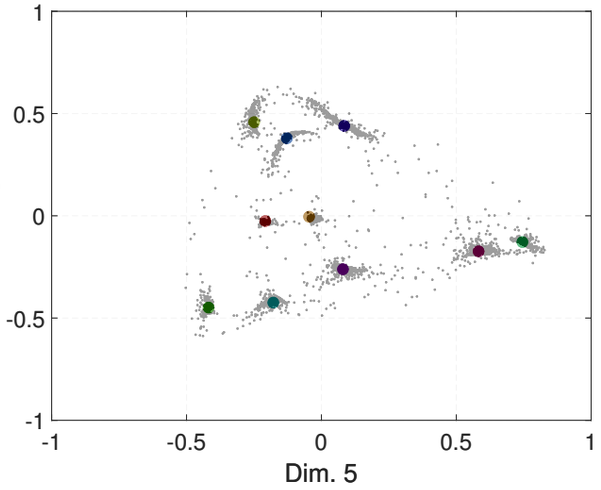
Summary

✓ Sketching framework

✓ Dimension reduction

✓ Empirical success





✓ Statistical guarantees
- ➡compressive PCA
- ➡compressive k-means
- ➡compressive GMM

✤ **Next challenges:**
- provably good recovery algorithms ?
- sketches for other learning tasks ?
- privacy guarantees ?

- toolbox sketchml.gforge.inria.fr
- preprint arxiv.org/abs/1706.07180

PLEASE

projection, learning and sparsity for efficient data processing