Sketched Clustering via Hybrid Approximate Message Passing

Evan Byrne, Rémi Gribonval, Philip Schniter

Sketched Clustering via Hybrid Approximate Message Passing

Evan Byrne,[*] Rémi Gribonval,[†] and Philip Schniter,[*]
[*]Dept. of ECE, The Ohio State Univ., Columbus, OH, 43210, USA. (byrne.133@osu.edu, schniter.1@osu.edu)
[†]Univ Rennes, Inria, CNRS, IRISA, France. (remi.gribonval@inria.fr)

***Abstract*—In sketched clustering, the dataset is first sketched down to a vector of modest size, from which the cluster centers are subsequently extracted. The goal is to perform clustering more efficiently than with methods that operate on the full training data, such as k-means++. For the sketching methodology recently proposed by Keriven, Gribonval, et al., which can be interpreted as a random sampling of the empirical characteristic function, we propose a cluster recovery algorithm based on simplified hybrid generalized approximate message passing (SHyGAMP). Numerical experiments suggest that our approach is more efficient than the state-of-the-art sketched clustering algorithms (in both computational and sample complexity) and more efficient than k-means++ in certain regimes.**

I. Introduction

Given a dataset $\boldsymbol{X} \triangleq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T] \in \mathbb{R}^{N\times T}$ comprising T feature vectors of dimension N, the standard clustering problem is to find K centroids $\boldsymbol{C} \triangleq [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K] \in \mathbb{R}^{N\times K}$ that minimize the sum of squared errors (SSE)

$$\text{SSE}(\boldsymbol{X}, \boldsymbol{C}) \triangleq \sum_{t=1}^{T} \min_k \|\boldsymbol{x}_t - \boldsymbol{c}_k\|_2^2. \tag{1}$$

Finding the optimal $\boldsymbol{C}$ is an NP-hard problem [1]. Thus, many heuristic approaches have been proposed, with one of the most popular being the *k-means algorithm* [2], [3]. Because k-means can get trapped in bad local minima, many robust variants have been proposed. One of the best known is *k-means++* [4], which uses a careful random initialization procedure to yield solutions with SSE that are on average $\leq 8(\ln K + 2)$ times the minimal SSE. But even with k-means++, many random re-initializations may be required to find a near-optimal clustering. For each initialization, the computational complexity of k-means++ scales as $O(TKNI)$, with I the number of iterations, which can be prohibitive for large T.

A. Sketched Clustering

In *sketched clustering* [5], [6], the dataset is first sketched down to a vector $\boldsymbol{y}$ with $M \ll TN$ components, from which the cluster centers are subsequently extracted. If the sketch can be performed efficiently, then—since the cluster-extraction complexity will be independent of T—there is a chance that sketched clustering will be more efficient than direct clustering methods like k-means++ when T is large.

[*]E. Byrne and P. Schniter acknowledge support from NSF grant 1716388 and MIT Lincoln Labs.

The approach proposed in [5], [6] uses a sketch $\boldsymbol{y} \triangleq [y_1, \ldots, y_M]^\mathsf{T}$ of the form

$$y_m = \frac{1}{T}\sum_{t=1}^{T} \exp(\mathrm{j}\boldsymbol{w}_m^\mathsf{T}\boldsymbol{x}_t) \tag{2}$$

with randomly generated $\boldsymbol{W} \triangleq [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M]^\mathsf{T} \in \mathbb{R}^{M\times N}$. Note that y_m in (2) can be interpreted as a sample of the empirical characteristic function, i.e.,

$$\phi_{\mathsf{x}}(\boldsymbol{w}_m) = \int_{\mathbb{R}^N} p_{\mathsf{x}}(\boldsymbol{x}) \exp(\mathrm{j}\boldsymbol{w}_m^\mathsf{T}\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \tag{3}$$

under the empirical distribution $p_{\mathsf{x}}(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T}\delta(\boldsymbol{x} - \boldsymbol{x}_t)$. Note that sketching $\boldsymbol{X}$ as $\boldsymbol{y}$ via (2) costs $O(TMN)$ operations, but it is easily parallelized.

For the recovery of cluster centers from $\boldsymbol{y}$, the state-of-the-art algorithm is *compressed learning orthogonal matching pursuit with replacement* (CL-OMPR) [5], [6]. It aims to solve

$$\underset{\boldsymbol{C},\boldsymbol{\alpha}}{\arg\min} \sum_{m=1}^{M} \left| y_m - \sum_{k=1}^{K} \alpha_k \exp(\mathrm{j}\boldsymbol{w}_m^\mathsf{T}\boldsymbol{c}_k) \right|^2 \tag{4}$$

using a greedy heuristic inspired by the *orthogonal matching pursuit* algorithm [7] popular in compressed sensing. With $M \approx 10KN$, CL-OMPR often attains SSEs similar to those attained with k-means++, despite the lack of a direct link between the problem formulations (1) and (4). CL-OMPR's computational complexity is $O(MNK^2)$, however, which can be impractical when K is large. Thus, we seek a sketched clustering scheme whose complexity grows linearly in K.

B. Contributions

For the recovery of the cluster centers from a sketch $\boldsymbol{y}$ of the form given in (2), we propose *compressive learning via approximate message passing* (CL-AMP). As we will see, CL-AMP has a computational complexity of $O(MNK)$ and performs favorably to CL-OMPR in terms of both runtime and sample complexity M. Furthermore, we find that CL-AMP performs favorably to k-means++ in certain operating regimes. CL-AMP can be understood as an application of the *simplified hybrid generalized approximate message passing* (SHyGAMP) framework [8] to sketched clustering. Further details will be provided in the sequel.

II. Compressive Learning via AMP

A. High-Dimensional Inference Framework

CL-AMP formulates cluster recovery as a high-dimensional inference problem rather than as an optimization problem like

(4). In particular, it assumes a Gaussian mixture model (GMM)

$$\boldsymbol{x}_t \sim \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{c}_k, \boldsymbol{R}_k). \tag{5}$$

where the GMM means are the cluster centers $\boldsymbol{c}_k$ and the GMM weights α_k and covariances $\boldsymbol{R}_k$ are unknown.

Defining $g_m \triangleq \|\boldsymbol{w}_m\|$ and $\widetilde{\boldsymbol{w}}_m \triangleq \boldsymbol{w}_m/g_m$, so $\|\widetilde{\boldsymbol{w}}_m\| = 1$,

$$y_m = \frac{1}{T}\sum_{t=1}^{T} \exp(\mathrm{j}\boldsymbol{w}_m^\mathsf{T}\boldsymbol{x}_t) \approx \mathrm{E}\{\exp(\mathrm{j}\boldsymbol{w}_m^\mathsf{T}\boldsymbol{x}_t)\} \tag{6}$$

$$= \sum_{k=1}^{K} \alpha_k \exp\Big(\mathrm{j}g_m \underbrace{\widetilde{\boldsymbol{w}}_m^\mathsf{T}\boldsymbol{c}_k}_{\triangleq\, z_{mk}} - g_m^2 \underbrace{\widetilde{\boldsymbol{w}}_m^\mathsf{T}\boldsymbol{R}_k\widetilde{\boldsymbol{w}}_m}_{\triangleq\, \tau_{mk}}/2\Big), \tag{7}$$

where (6) holds under large T and (7) follows from the facts that $\boldsymbol{w}_m^\mathsf{T}\boldsymbol{x}_t \sim \sum_k \alpha_k \mathcal{N}(g_m z_{mk}, g_m^2\tau_{mk})$ and that $\mathrm{E}\{e^{\mathrm{j}x}\} = e^{\mathrm{j}z-\tau/2}$ when $x \sim \mathcal{N}(z,\tau)$. For $\widetilde{\boldsymbol{w}}_m$ uniform on the sphere,

$$\tau_{mk} \xrightarrow{p} \mathrm{E}\{\tau_{mk}\} = \mathrm{tr}(\boldsymbol{R}_k)/N \triangleq \tau_k \tag{8}$$

as $N \to \infty$, as long as the peak-to-average eigenvalue ratio of $\boldsymbol{R}_k$ remains bounded [9]. Thus, for $\boldsymbol{z}_m \triangleq [z_{m1}, \dots, z_{mK}]^\mathsf{T}$,

$$p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}_m;\boldsymbol{\alpha},\boldsymbol{\tau}) = \delta\Bigg(y_m - \sum_{k=1}^{K}\alpha_k \exp\Big(\mathrm{j}g_m z_{mk} - \frac{g_m^2\tau_k}{2}\Big)\Bigg) \tag{9}$$

with hyperparameters $\boldsymbol{\tau} \triangleq [\tau_1, \dots, \tau_K]^\mathsf{T}$, $\boldsymbol{\alpha} \triangleq [\alpha_1, \dots, \alpha_K]^\mathsf{T}$.

The cluster coordinates $\{c_{nk}\}$ are modeled as i.i.d. $p_\mathsf{c}(c;\nu)$, where nominally $p_\mathsf{c}(c;\nu) = \mathcal{N}(c;0,\nu)$ with large ν. Our main objective is then to compute the conditional mean

$$\widehat{\boldsymbol{C}} = \mathrm{E}\{\mathbf{C}\,|\,\boldsymbol{y}\}, \tag{10}$$

where the expectation is taken over

$$p(\boldsymbol{C}|\boldsymbol{y}) \propto \prod_{m=1}^{M} p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{w}_m^\mathsf{T}\boldsymbol{C};\boldsymbol{\alpha},\boldsymbol{\tau}) \prod_{k=1}^{K}\prod_{n=1}^{N} p_\mathsf{c}(c_{nk};\nu), \tag{11}$$

while simultaneously learning the hyperparameters $\boldsymbol{\alpha}, \boldsymbol{\tau}, \nu$. Here and in the sequel, we format random variables in san-serif font for clarity.

B. Approximate Message Passing

Exact computation of $\widehat{\boldsymbol{C}}$ in (10) is impractical due to the form of $p_{\mathsf{y}|\mathbf{z}}$. One might consider approximate inference via the sum-product algorithm (SPA), but even the SPA is intractable due to the form of $p_{\mathsf{y}|\mathbf{z}}$. Given the presence of a large random matrix $\boldsymbol{W}$ in the problem formulation, we instead proposed to tackle approximate inference using *approximate message passing* (AMP) [10]. In particular, we apply the *simplified hybrid generalized AMP* (SHyGAMP) methodology from [8], while simultaneously estimating $\boldsymbol{\alpha}, \boldsymbol{\tau}, \nu$ through expectation maximization (EM). Some background on AMP methods will now be provided to justify our approach.

The original AMP algorithm of Donoho, Maleki, and Montanari [10] was designed to estimate i.i.d. $\boldsymbol{c}$ under the standard linear model (i.e., $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{c} + \boldsymbol{n}$ with known $\boldsymbol{W} \in \mathbb{R}^{M\times N}$ and additive white Gaussian noise $\boldsymbol{n}$). The generalized AMP (GAMP) algorithm of Rangan [11] extended AMP to the generalized linear model (i.e., $\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{z})$ for $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{c}$ and separable $p(\boldsymbol{y}|\boldsymbol{z}) = \prod_{m=1}^{M} p(y_m|z_m)$). Both AMP and GAMP give accurate approximations of the SPA under large i.i.d. sub-Gaussian $\boldsymbol{W}$, while maintaining a computational complexity of only $O(MN)$. Furthermore, both can be rigorously analyzed via the state-evolution framework, which shows that they are Bayes-optimal in certain regimes [12].

A limitation of AMP [10] and GAMP [11] is that they cover only problems with i.i.d. estimand $\boldsymbol{c}$ and separable likelihood $p(\boldsymbol{y}|\boldsymbol{z}) = \prod_{m=1}^{M} p(y_m|z_m)$. Thus, Hybrid GAMP (HyGAMP) [13] was developed to tackle problems with a structured prior and/or likelihood. HyGAMP could be applied to (10)-(11), but it requires computing and inverting $O(N+M)$ covariance matrices of dimension K at each iteration. The SHyGAMP algorithm [8] is a simplification of HyGAMP that uses diagonal covariance matrices to drastically reduce complexity. As described in [8], SHyGAMP can be readily combined with the EM algorithm for hyperparameter learning.

C. SHyGAMP

The SHyGAMP algorithm is summarized in Algorithm 1 using the language of Section II-A, assuming $\widetilde{\boldsymbol{W}}$ has unit-norm rows. There, with some abuse of notation, we use $\boldsymbol{c}_n^\mathsf{T}$ to denote the nth row of $\boldsymbol{C}$ (where previously we used $\boldsymbol{c}_k$ to denote the kth column of $\boldsymbol{C}$). We also use $\widehat{\boldsymbol{P}} \triangleq [\widehat{\boldsymbol{p}}_1, \dots, \widehat{\boldsymbol{p}}_M]^\mathsf{T}$, $\widehat{\boldsymbol{Z}} \triangleq [\widehat{\boldsymbol{z}}_1, \dots, \widehat{\boldsymbol{z}}_M]^\mathsf{T}$, $\widehat{\boldsymbol{R}} \triangleq [\widehat{\boldsymbol{r}}_1, \dots, \widehat{\boldsymbol{r}}_N]^\mathsf{T}$, $\oslash$ for componentwise division, and $\odot$ for componentwise multiplication.

At each iteration, lines 10-11 of Algorithm 1 compute an approximation of the posterior mean and variance of $\{\mathsf{c}_{nk}\}$ using the "pseudo-measurements" $\widehat{\boldsymbol{r}}_n = \boldsymbol{c}_n + \boldsymbol{v}_n$, where $\boldsymbol{v}_n$ is treated as a typical realization of $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}^\mathbf{r})$. Thus, the approximate posterior pdf used in lines 10-11 is

$$p_{\mathbf{c}|\mathbf{r}}(\boldsymbol{c}_n|\widehat{\boldsymbol{r}}_n;\boldsymbol{Q}^\mathbf{r}) = \frac{p_\mathbf{c}(\boldsymbol{c}_n)\mathcal{N}(\boldsymbol{c}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}^\mathbf{r})}{\int p_\mathbf{c}(\boldsymbol{c}_n')\mathcal{N}(\boldsymbol{c}_n';\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}^\mathbf{r})\,\mathrm{d}\boldsymbol{c}_n'}. \tag{12}$$

Similarly, lines 4-5 approximate the posterior mean and covariance of $\mathbf{z}_m \triangleq \mathbf{C}^\mathsf{T}\boldsymbol{w}_m$, which uses the pseudo-prior $\mathbf{z}_m \sim \mathcal{N}(\widehat{\boldsymbol{p}}_m, \boldsymbol{Q}^\mathbf{p})$ and hence the approximate posterior pdf

$$p_{\mathbf{z}|\mathsf{y},\mathbf{p}}(\boldsymbol{z}_m|y_m,\widehat{\boldsymbol{p}}_m;\boldsymbol{Q}^\mathbf{p}) = \frac{p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^\mathbf{p})}{\int p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}_m')\mathcal{N}(\boldsymbol{z}_m';\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^\mathbf{p})\,\mathrm{d}\boldsymbol{z}_m'}. \tag{13}$$

Essentially, the SHyGAMP algorithm breaks an NK-dimensional inference problem into $M+N$ K-dimensional inference problems involving an independent-Gaussian pseudo-prior or pseudo-likelihood, evaluated iteratively. The resulting computational complexity is $O(MNK)$.

D. From SHyGAMP to CL-AMP

The SHyGAMP algorithm can be applied to many different problems via appropriate choice of $p_{\mathsf{y}|\mathbf{z}}$ and $p_\mathbf{c}$. To apply SHyGAMP to sketched-clustering, we choose $p_{\mathsf{y}|\mathbf{z}}$ and $p_\mathbf{c}$ as described in Section II-A. The principal remaining challenge is to evaluate lines 4-5 of Algorithm 1.

Algorithm 1 SHyGAMP

```
Require: Measurements $\boldsymbol{y}$, matrix $\widetilde{\boldsymbol{W}}$ with $\|\widetilde{\boldsymbol{W}}\|_F^2 = M$, pdfs $p_{\mathsf{c}|\mathsf{r}}$ and
    $p_{\mathsf{z}|\mathsf{y},\mathsf{p}}$ from (12)-(13), initializations $\widehat{\boldsymbol{C}} = \mathrm{E}\{\mathsf{C}\}$, $\boldsymbol{q}_n^{\mathsf{c}} = \mathrm{diag}(\mathrm{cov}\{\mathsf{c}_n\})$
Ensure: $\widehat{\boldsymbol{S}} \leftarrow \mathbf{0}$.
 1: repeat
 2:     $\boldsymbol{q}^{\mathsf{p}} \leftarrow \frac{1}{N}\sum_{n=1}^N \boldsymbol{q}_n^{\mathsf{c}}$
 3:     $\widehat{\boldsymbol{P}} \leftarrow \widetilde{\boldsymbol{W}}\widehat{\boldsymbol{C}} - \widehat{\boldsymbol{S}}\,\mathrm{Diag}(\boldsymbol{q}^{\mathsf{p}})$
 4:     $\boldsymbol{q}_m^{\mathsf{z}} \leftarrow \mathrm{diag}\left(\mathrm{cov}\left\{\mathsf{z}_m \,\middle|\, y_m, \mathsf{p}_m = \widehat{\boldsymbol{p}}_m; \mathrm{Diag}(\boldsymbol{q}^{\mathsf{p}})\right\}\right) \quad \forall m$
 5:     $\widehat{\boldsymbol{z}}_m \leftarrow \mathrm{E}\left\{\mathsf{z}_m \,\middle|\, y_m, \mathsf{p}_m = \widehat{\boldsymbol{p}}_m; \mathrm{Diag}(\boldsymbol{q}^{\mathsf{p}})\right\} \quad \forall m$
 6:     $\boldsymbol{q}^{\mathsf{s}} \leftarrow \mathbf{1} \oslash \boldsymbol{q}^{\mathsf{p}} - \left(\frac{1}{M}\sum_{m=1}^M \boldsymbol{q}_m^{\mathsf{z}}\right) \oslash (\boldsymbol{q}^{\mathsf{p}} \odot \boldsymbol{q}^{\mathsf{p}})$
 7:     $\widehat{\boldsymbol{S}} \leftarrow (\widehat{\boldsymbol{Z}} - \widehat{\boldsymbol{P}})\,\mathrm{Diag}(\boldsymbol{q}^{\mathsf{p}})^{-1}$
 8:     $\boldsymbol{q}^{\mathsf{r}} \leftarrow \frac{N}{M}\mathbf{1} \oslash \boldsymbol{q}^{\mathsf{s}}$
 9:     $\widehat{\boldsymbol{R}} \leftarrow \widehat{\boldsymbol{C}} + \widetilde{\boldsymbol{W}}\widehat{\boldsymbol{S}}^{\mathsf{T}}\,\mathrm{Diag}(\boldsymbol{q}^{\mathsf{r}})$
10:     $\boldsymbol{q}_n^{\mathsf{c}} \leftarrow \mathrm{diag}\left(\mathrm{cov}\left\{\mathsf{c}_n \,\middle|\, \mathsf{r}_n = \widehat{\boldsymbol{r}}_n; \mathrm{Diag}(\boldsymbol{q}^{\mathsf{r}})\right\}\right) \quad \forall n$
11:     $\widehat{\boldsymbol{c}}_n \leftarrow \mathrm{E}\left\{\mathsf{c}_n \,\middle|\, \mathsf{r}_n = \widehat{\boldsymbol{r}}_n; \mathrm{Diag}(\boldsymbol{q}^{\mathsf{r}})\right\} \quad \forall n$
12: until Terminated
```

1) Inference of z_m: For lines 4-5 of Algorithm 1, we would like to compute the mean and variance

$$\widehat{z}_{mk} = C_m^{-1}\int_{\mathbb{R}^K} z_{mk}\, p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)\mathcal{N}\big(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}^{\mathsf{p}}\big)\,\mathrm{d}\boldsymbol{z}_m \quad (14)$$

$$q_{mk}^{\mathsf{z}} = \frac{\int_{\mathbb{R}^K}(z_{mk} - \widehat{z}_{mk})^2 p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)\mathcal{N}\big(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}^{\mathsf{p}}\big)\,\mathrm{d}\boldsymbol{z}_m}{C_m}, \quad (15)$$

where $C_m = \int_{\mathbb{R}^K} p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)\mathcal{N}\big(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}^{\mathsf{p}}\big)\,\mathrm{d}\boldsymbol{z}_m$. We propose approximations of $\widehat{z}_{mk}$ and q_{mk}^{z} that are summarized below; a full derivation has been omitted due to space limitations. For the remainder of this section, we omit the subscripts m and $\mathsf{y}|\mathsf{z}$ to simplify the notation.

The main idea behind our approximation of $\widehat{z}_{mk}$ and q_{mk}^{z} is to define $\theta_k \triangleq g z_k$ and then apply the Gaussian approximation (whose accuracy grows with K)

$$p\Big(\begin{bmatrix}\mathrm{Re}\{y\}\\ \mathrm{Im}\{y\}\end{bmatrix}\Big|\theta_k\Big) \approx \mathcal{N}\Big(\begin{bmatrix}\mathrm{Re}\{y\}\\ \mathrm{Im}\{y\}\end{bmatrix}; \beta_k\begin{bmatrix}\cos(\theta_k)\\ \sin(\theta_k)\end{bmatrix} + \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\Big) \quad (16)$$

to (9), where

$$\boldsymbol{\mu}_k = \sum_{l\neq k}\alpha_l e^{-g^2(\tau_k + [\boldsymbol{Q}^{\mathsf{p}}]_{kk})/2}\begin{bmatrix}\cos(g\widehat{p}_l)\\ \sin(g\widehat{p}_l)\end{bmatrix} \quad (17)$$

$$\boldsymbol{\Sigma}_k = \frac{1}{2}\sum_{l\neq k}\beta_l^2\big(1 - e^{-g^2[\boldsymbol{Q}^{\mathsf{p}}]_{ll}}\big) \times \Big(\boldsymbol{I} - e^{-g^2[\boldsymbol{Q}^{\mathsf{p}}]_{ll}}\begin{bmatrix}\cos(2g\widehat{p}_l) & \sin(2g\widehat{p}_l)\\ \sin(2g\widehat{p}_l) & -\cos(2g\widehat{p}_l)\end{bmatrix}\Big) \quad (18)$$

$$\beta_k = \alpha_k\exp(-g^2\tau_k/2). \quad (19)$$

Rewriting (16) as

$$p\Big(\beta_k^{-1}\begin{bmatrix}\mathrm{Re}\{y\}\\ \mathrm{Im}\{y\}\end{bmatrix}\Big|\theta_k\Big) \approx \mathcal{N}\Big(\begin{bmatrix}\cos(\theta_k)\\ \sin(\theta_k)\end{bmatrix}; \beta_k^{-1}\begin{bmatrix}\mathrm{Re}\{y\}\\ \mathrm{Im}\{y\}\end{bmatrix} - \beta_k^{-1}\boldsymbol{\mu}_k, \beta_k^{-2}\boldsymbol{\Sigma}_k\Big), \quad (20)$$

the right side of (20) can be recognized as being proportional to the *generalized von Mises* (GvM) density [14] over $\theta_k \in [0, 2\pi)$. Under this GvM approximation, we have [14] that

$$p(y|\theta_k) \propto \exp\big(\kappa_k\cos(\theta_k - \zeta_k) + \bar{\kappa}_k\cos[2(\theta_k - \bar{\zeta}_k)]\big) \quad (21)$$

for parameters $\kappa_k, \bar{\kappa}_k > 0$ and $\zeta_k, \bar{\zeta}_k \in [0, 2\pi)$ defined from y, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, and β_k. In particular,

$$\kappa_k\cos(\zeta_k) = -\frac{1}{1-\rho_k^2}\Big(\frac{\rho_k\bar{\nu}_k}{\sigma_k\bar{\sigma}_k} - \frac{\nu_k}{\sigma_k^2}\Big) \quad (22)$$

$$\kappa_k\sin(\zeta_k) = -\frac{1}{1-\rho_k^2}\Big(\frac{\rho_k\nu_k}{\sigma_k\bar{\sigma}_k} - \frac{\bar{\nu}_k}{\bar{\sigma}_k^2}\Big) \quad (23)$$

$$\bar{\kappa}_k\cos(2\bar{\zeta}_k) = -\frac{1}{4(1-\rho_k^2)}\Big(\frac{1}{\sigma_k^2} - \frac{1}{\bar{\sigma}_k^2}\Big) \quad (24)$$

$$\bar{\kappa}_k\sin(2\bar{\zeta}_k) = \frac{\rho_k}{2(1-\rho_k^2)\sigma_k\bar{\sigma}_k}, \quad (25)$$

where

$$\begin{bmatrix}\nu_k\\ \bar{\nu}_k\end{bmatrix} \triangleq \beta_k^{-1}\Big(\begin{bmatrix}\mathrm{Re}\{y\}\\ \mathrm{Im}\{y\}\end{bmatrix} - \boldsymbol{\mu}_k\Big) \quad (26)$$

and

$$\begin{bmatrix}\sigma_k^2 & \rho_k\sigma_k\bar{\sigma}_k\\ \rho_k\sigma_k\bar{\sigma}_k & \bar{\sigma}_k^2\end{bmatrix} \triangleq \beta_k^{-2}\boldsymbol{\Sigma}_k. \quad (27)$$

Given the SHyGAMP pseudo-prior $\mathsf{z}_k \sim \mathcal{N}(\widehat{p}_k, [\boldsymbol{Q}^{\mathsf{p}}]_{kk})$, the posterior on θ_k takes the form

$$p(\theta_k|y) \propto \mathcal{N}\big(\theta_k; g\widehat{p}_k, g^2[\boldsymbol{Q}^{\mathsf{p}}]_{kk}\big)\,p(y|\theta_k) \quad (28)$$
$$\propto \exp\Big[\kappa_k\cos(\theta_k - \zeta_k) + \bar{\kappa}_k\cos[2(\theta_k - \bar{\zeta}_k)] - \frac{(\theta_k - g\widehat{p}_k)^2}{2g^2[\boldsymbol{Q}^{\mathsf{p}}]_{kk}}\Big].$$

We then face the task of computing $\mathrm{E}\{\theta_k|y\}$ and $\mathrm{E}\{\theta_k^2|y\}$ under (28). Several methods could be applied here, such as numerical integration. The method employed for the experiments in Section III is based on the Laplace approximation [15]. For this, we compute $\widehat{\theta}_{k,\mathsf{MAP}} \triangleq \arg\max_{\theta_k}\ln p(\theta_k|y)$ using bisection and then approximate $\mathrm{E}\{\theta_k|y\} \approx \widehat{\theta}_{k,\mathsf{MAP}}$ and $\mathrm{var}\{\theta_k|y\} \approx -\frac{\mathrm{d}^2}{\mathrm{d}\theta_k^2}\ln p(\theta_k|y)\big|_{\theta_k = \widehat{\theta}_{k,\mathsf{MAP}}}$. Finally, we compute $\widehat{z}_k = \mathrm{E}\{\theta_k|y\}/g$ and $q_k^{\mathsf{z}} = \mathrm{var}\{\theta_k|y\}/g^2$.

2) Inference of c_n: For lines 10-11 of Algorithm 1, recall that $p_{\mathsf{c}}(\boldsymbol{c}_n) = \mathcal{N}(\boldsymbol{c}_n; \mathbf{0}, \nu\boldsymbol{I})$. Thus $p_{\mathsf{c}|\mathsf{r}}$ is Gaussian and the posterior mean and covariance of c_n can be computed straightforwardly as

$$\boldsymbol{Q}_n^{\mathsf{c}} = \big(\nu^{-1}\boldsymbol{I} + [\boldsymbol{Q}^{\mathsf{r}}]^{-1}\big)^{-1} \triangleq \boldsymbol{Q}^{\mathsf{c}} \quad (29)$$

$$\widehat{\boldsymbol{c}}_n = \boldsymbol{Q}^{\mathsf{c}}[\boldsymbol{Q}^{\mathsf{r}}]^{-1}\widehat{\boldsymbol{r}}_n \quad (30)$$

Above, $[\boldsymbol{Q}^{\mathsf{r}}]^{-1}$ is simplified by the fact that $\boldsymbol{Q}^{\mathsf{r}}$ is diagonal.

E. Hyperparameter Tuning

The likelihood model $p_{\mathsf{y}|\mathsf{z}}$ in (9) depends on the unknown hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$. Similarly, the prior p_{c} depends on the unknown variance ν. We propose to estimate these hyperparameters using a combination of *expectation maximization* (EM) and SHyGAMP, as suggested in [8] and detailed—for the simpler case of GAMP—in [16]. Extrapolating [16] to the SHyGAMP case, we estimate $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$ via

$$\{\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\tau}}\} = \operatorname*{arg\,max}_{\boldsymbol{\alpha}\geq\mathbf{0}, \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{1}=1, \boldsymbol{\tau}>\mathbf{0}}\sum_{m=1}^M\int_{\mathbb{R}^K}\mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{z}}_m, \mathrm{Diag}(\boldsymbol{q}_m^{\mathsf{z}})) \times \ln p(y_m|\boldsymbol{z}_m; \boldsymbol{\alpha}, \boldsymbol{\tau})\,\mathrm{d}\boldsymbol{z}_m \quad (31)$$

at each SHyGAMP iteration, immediately after line 5 in Algorithm 1. For tractability, we approximate the Dirac delta in (9) by a Gaussian pdf with small variance $\epsilon > 0$, giving

$$\ln p(y_m|\boldsymbol{z}_m;\boldsymbol{\alpha},\boldsymbol{\tau}) \approx -\frac{1}{\epsilon}\left|y - \sum_{k=1}^{K}\alpha_k \exp\left(\mathrm{j}g_m z_{mk} - \frac{g_m^2\tau_k}{2}\right)\right|^2 + \text{const.} \tag{32}$$

The resulting optimization problem (31) (which does not depend on ϵ) can be straightforwardly solved using gradient projection, since closed-form expressions for the objective and its gradient exist.

III. Numerical Experiments

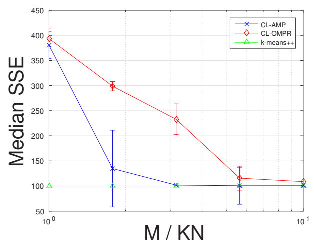
In this section, we present the results of two numerical experiments used to test the performance of the CL-AMP, CL-OMPR, and k-means++ algorithms. For k-means++, we used the implementation provided by MATLAB and, for CL-OMPR, we downloaded the MATLAB implementation from [17] and enabled the "++" initialization method. CL-OMPR and CL-AMP used the same sketch $\boldsymbol{y}$, whose frequency vectors $\boldsymbol{W}$ were drawn using the method described in [5]. For both experiments, the clusters were randomly drawn as $\boldsymbol{c}_k \sim \mathcal{N}(\boldsymbol{0}_N, 1.5^2 K^{2/N}\boldsymbol{I}_N)$, after which the training (and test) data were drawn from the GMM (5) with weights $\alpha_k = \frac{1}{K}\forall k$ and covariances $\boldsymbol{R}_k = \boldsymbol{I}_N \forall k$. For CL-OMPR and CL-AMP, the runtimes reported include the time of computing the sketch.

A. SSE Minimization

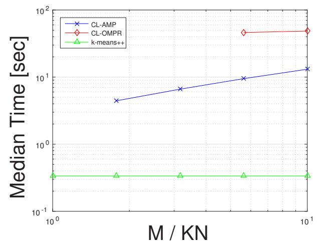
In the first experiment, we test each algorithm's ability to minimize SSE on the training data, i.e., to solve the problem (1). For each pair of $(K,N) \in \{(5,100),(10,50),(10,100)\}$, 10 trials were performed, where in each trial, a training dataset was randomly generated with $T = 10^4$ samples. For cluster recovery, k-means++ was invoked on this training dataset with 1 replicate (i.e., 1 run from a random initialization), while CL-AMP and CL-OMPR were invoked using a sketch of length M. Several values of M, logarithmically spaced in the interval $[KN, 10KN]$, were evaluated.

For each (K,N) pair under test, Figs. 1a, 1c, and 1e show the median SSE of CL-AMP and CL-OMPR versus M/KN, with the error-bars showing the standard deviation. The median SSE of k-means++ was superimposed on these figures as a reference, although k-means++ has no dependence on M. Likewise, Figs. 1b, 1d, and 1f show the corresponding median runtime for CL-AMP and CL-OMPR vs M/KN, where again the result for k-means++ was superimposed. Because a low runtime is meaningless if the corresponding SSE is very high, the runtime was not shown for CL-AMP or CL-OMPR when its SSE was more than twice that of k-means++.

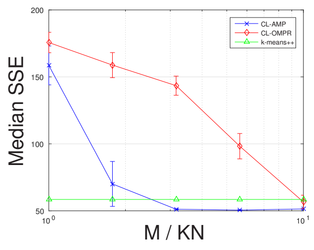
Figure 1 shows that CL-AMP achieved a low SSE with fewer measurements M than CL-OMPR. In particular, CL-AMP required $M \approx 3KN$ to minimize the SSE, while CL-OMPR required $M \approx 10KN$. Also, the minimum SSE achieved by CL-AMP was in most cases lower than that of k-means++ and CL-OMPR. As we will see in Fig. 2, the SSE performance of k-means++ can be improved (at the expense of runtime) by increasing the number of replicates. Figure 1 also shows that CL-AMP was an order of magnitude faster than CL-OMPR for all (K,N,M) under test. Meanwhile, CL-AMP was approximately one order-of-magnitude slower than k-means++, although the SSE achieved by CL-AMP was often lower. A direct SSE-vs-runtime comparison is given below.



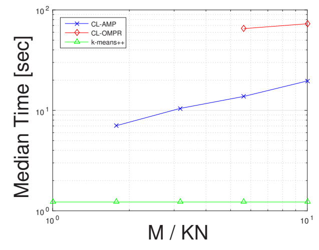
(a) $K=5$, $N=100$



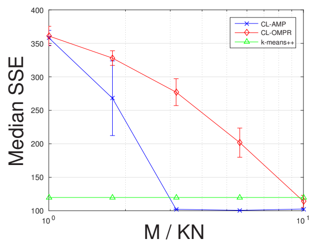
(b) $K=5$, $N=100$



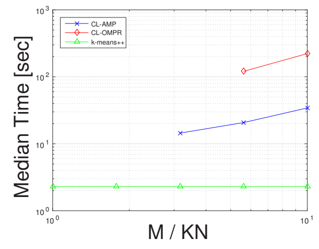
(c) $K=10$, $N=50$



(d) $K=10$, $N=50$



(e) $K=10$, $N=100$



(f) $K=10$, $N=100$

Fig. 1: Median sum-squared error and runtime vs M/KN.

B. Performance versus Runtime

The previous experiment demonstrated CL-AMP's ability to minimize SSE faster and with fewer measurements than CL-OMPR. However, the comparison with k-means++ was inconclusive: k-means++ was faster but achieved a worse SSE in many cases. We now describe a different experiment that aimed to evaluate clustering performance versus runtime. For clustering performance, we consider both training SSE and classification error on test data. For the latter, training data is used for cluster recovery and the estimated clusters are used for minimum-distance classification of test data. To control the performance and computational complexity of k-means++, we allowed multiple replicates as well as training-data subsampling. Details are provided in the sequel.

We first drew $K = 30$ random centroids of dimension $N = 20$ under the previously described GMM. Then we

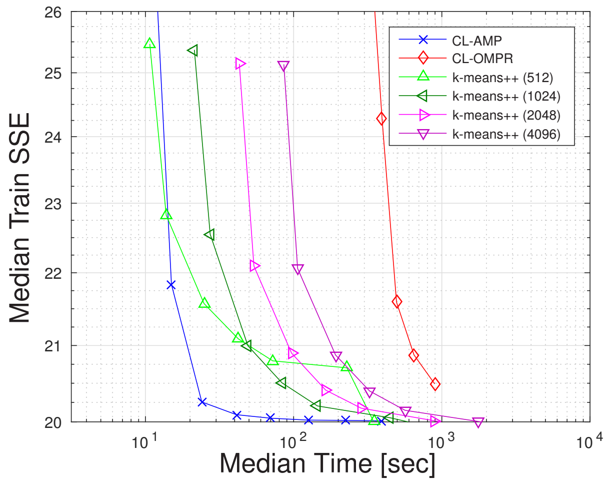


Fig. 2: Training sum-squared error vs runtime. Each k-means++ trace corresponds to a different number of replicates.

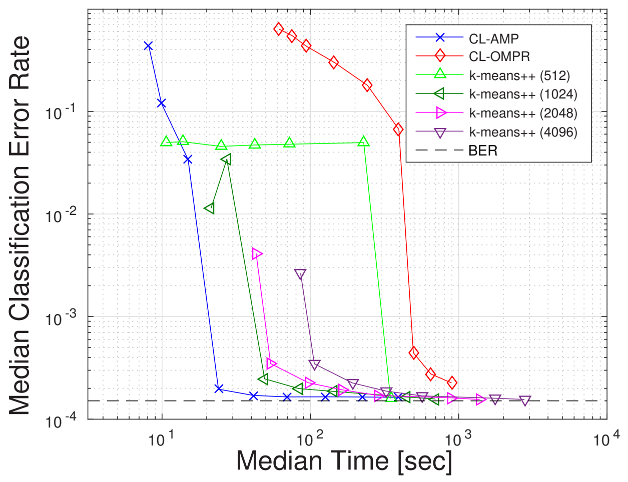


Fig. 3: Classification error rate vs runtime. Each k-means++ trace corresponds to a different number of replicates.

generated $T = 10^4$ random training samples from this GMM. To recover clusters, CL-AMP and CL-OMPR were applied with sketch length M, while k-means++ was applied with random subsampling of the training set and multiple replicates. We tested several sketch lengths $M \in [KN, 100KN]$, k-means++ sampling rates $\in [0.5^6, 1]$, and k-means++ replicates $\in [512, 4096]$, all logarithmically spaced. Finally, the resulting training-data SSE was evaluated using the full training dataset.

The quality of the estimated centroids was also evaluated by computing the error-rate of minimum-distance classification of a test dataset (of size $T_{\text{test}} = 5 \times 10^6$, drawn from the same GMM as the training data). Here, we used the Hungarian algorithm to assign labels to the estimated centroids.

Figure 2 shows median training SSE versus runtime over 10 trials for each algorithm under test, while Fig. 3 shows the corresponding median test error rate versus runtime. In each CL-AMP and CL-OMPR trace, the different datapoints correspond to increasing values of sketch length M, while in each k-means++ trace, the different datapoints correspond to increasing sampling rates for a fixed number of replicates (specified in the legend). Figure 3 shows the corresponding classification error rate, computed on the test set, as well as the Bayes (i.e., minimum possible) classification error rate.

Figures 2 and 3 tell a similar story: to achieve near-optimal training-SSE or classification error-rate with this GMM, CL-AMP (with properly adjusted M) requires less runtime than CL-OMPR or any variation of k-means++. Note that CL-AMP is easily "tuned" by choosing $M \approx 5KN$, while k-means++ is much more difficult to tune: it is not clear how to choose the best combination of subsampling rate and number-of-replicates to achieve both low SSE and low runtime. Note also that the implementation of k-means++ is highly optimized while that of CL-AMP is not, so further improvements may be possible by optimizing the implementation of CL-AMP.

References

[1] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, "Clustering large graphs via the singular value decomposition," *Mach. Learn.*, vol. 56, no. 1-3, pp. 9–33, 2004.

[2] H. Steinhaus, "Sur la division des corps matériels en parties," *Bull. Acad. Polon. Sci.*, vol. 4, no. 12, pp. 801–804, 1956.

[3] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, pp. 651–666, June 2010.

[4] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. ACM-SIAM Symp. Discrete Alg.*, pp. 1027–1035, 2007.

[5] N. Keriven, A. Bourrier, R. Gribonval, and P. Perez, "Sketching for large-scale learning of mixture models," Jun 2016. (found at *arXiv:1606.02838*).

[6] N. Keriven, N. Tremblay, Y. Traonmilin, and R. Gribonval, "Compressive k-means," Oct 2016. (found at *arXiv:1610.08738*).

[7] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. Asilomar Conf. Signals Syst. Comput.*, (Pacific Grove, CA), pp. 40–44, 1993.

[8] E. M. Byrne and P. Schniter, "Sparse multinomial logistic regression via approximate message passing," *IEEE Trans. Signal Process.*, vol. 64, no. 21, pp. 5485–5498, 2016.

[9] M. Rudelson and R. Vershynin, "Hanson-Wright inequality and sub-Gaussian concentration," *Electron. Commun. Probab.*, vol. 18, no. 82, pp. 1–9, 2013.

[10] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, pp. 18914–18919, Nov. 2009.

[11] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int. Symp. Inform. Thy.*, pp. 2168–2172, Aug. 2011. (full version at *arXiv:1010.5141*).

[12] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inform. Theory*, vol. 57, pp. 764–785, Feb. 2011.

[13] S. Rangan, A. K. Fletcher, V. K. Goyal, E. Byrne, and P. Schniter, "Hybrid approximate message passing," *IEEE Trans. Signal Process.*, vol. 65, no. 17, pp. 4577–4592, 2017.

[14] R. Gatto and S. R. Jammalamadaka, "The generalized von Mises distribution," *Stat. Method.*, vol. 4, pp. 341–353, 2007.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2007.

[16] J. P. Vila and P. Schniter, "Expectation-maximization Gaussian-mixture approximate message passing," *IEEE Trans. Signal Process.*, vol. 61, pp. 4658–4672, Oct. 2013.

[17] N. Keriven, N. Tremblay, and R. Gribonval, "SketchMLbox : a Matlab toolbox for large-scale learning of mixture models," 2016. http://sketchml.gforge.inria.fr.