Supplementary Appendix

A Proof of Proposition 1

Here we provide for completeness the proof of Prop. 1 following Lücke and Forster [2017]. First, note that the free energy Eq. (6) is maximized w.r.t. $\hat{\Theta}$ by setting $\hat{\Theta} = \Theta$. This can be shown [Lücke, 2016] by generalizing a similar result for full posteriors [Neal and Hinton, 1998]. After setting $\hat{\Theta} = \Theta$ the free energy $\mathcal{F}(\mathcal{K}, \Theta, \Theta) =: \mathcal{F}(\mathcal{K}, \Theta)$ can be simplified to take on the following form [Lücke, 2016]:

$$\mathcal{F}(\mathcal{K}, \Theta) = \sum_n \log\Big(\sum_{c \in \mathcal{K}^{(n)}} p(c, \vec{y}^{\,(n)} | \, \Theta) \Big) \,. \tag{10}$$

Let us now repeat Prop. 1 in more detail before we reiterate the proof:

Proposition 1
Consider the GMM of Eq. (2) and the free energy Eq. (10) for $n = 1 : N$ data points $\vec{y}^{(n)} \in \mathbb{R}^D$. Furthermore, consider for a fixed n the replacement of a cluster $c \in \mathcal{K}^{(n)}$ by a cluster $\tilde{c} \notin \mathcal{K}^{(n)}$. Then the free energy $\mathcal{F}(\mathcal{K}, \Theta)$ increases if and only if

$$\|\vec{y}^{\,(n)} - \vec{\mu}_{\tilde{c}}\| < \|\vec{y}^{\,(n)} - \vec{\mu}_c\| \,. \tag{11}$$

Proof
By considering the specific functional form of Eq. (10) we can observe that the free energy is increased by replacing $c \in \mathcal{K}^{(n)}$ with $\tilde{c} \notin \mathcal{K}^{(n)}$ if $p(\tilde{c}, \vec{y}^{\,(n)} | \, \Theta) > p(c, \vec{y}^{\,(n)} | \, \Theta)$. This applies because of the summation over c in Eq. (10) and because of the concavity of the logarithm. Analogously, the free energy stays constant or decreases for $p(\tilde{c}, \vec{y}^{\,(n)} | \, \Theta) \leq p(c, \vec{y}^{\,(n)} | \, \Theta)$. If we insert for the joint probability $p(\tilde{c}, \vec{y} \,| \, \Theta)$ the GMM (2), we obtain:

$$p(\tilde{c}, \vec{y} \,| \, \Theta) = \tfrac{1}{C} (2\pi\sigma^2)^{-\frac{D}{2}} \exp\big(-\tfrac{1}{2\sigma^2} \|\vec{y} - \vec{\mu}_{\tilde{c}}\|^2\big). \tag{12}$$

The first two factors are independent of the data point and cluster. The criterion for an increase of the free energy can therefore be reformulated as follows:

$$\begin{aligned}
& p(\tilde{c}, \vec{y} \,| \, \Theta) > p(c, \vec{y} \,| \, \Theta) \\
\Leftrightarrow \quad & \exp\big(-\frac{1}{2\sigma^2} \|\vec{y} - \vec{\mu}_{\tilde{c}}\|^2\big) > \exp\big(-\frac{1}{2\sigma^2} \|\vec{y} - \vec{\mu}_c\|^2\big) \\
\Leftrightarrow \quad & -\frac{1}{2\sigma^2} \|\vec{y} - \vec{\mu}_{\tilde{c}}\|^2 > -\frac{1}{2\sigma^2} \|\vec{y} - \vec{\mu}_c\|^2 \\
\Leftrightarrow \quad & \|\vec{y} - \vec{\mu}_{\tilde{c}}\| < \|\vec{y} - \vec{\mu}_c\| \,.
\end{aligned} \tag{13}$$

□

B Illustration of Algorithms 1 and 2

The four sub-figures of Fig. B.1 illustrate how the variational E-steps of Alg. 1 find clusters increasingly close to $\vec{y}^{\,(n)}$. We use $C' = 3$ and $G = 5$ for this example (the same as for Fig. 1). For illustration purposes, we assume that good cluster centers have already been found and that they remain fixed across iterations. **A** The subfigure replicates Fig. 1, i.e., it shows for one data point $\vec{y}^{\,(n)}$ its set $\mathcal{K}^{(n)}$ and the sets $\mathcal{G}_c$ for all $c \in \mathcal{K}^{(n)}$. The $\mathcal{G}_c$ of all clusters are computed in the first block of Alg. 1. The $\mathcal{G}^{(n)}$ are (in the second block of Alg. 1) defined as the union over all $c \in \mathcal{K}^{(n)}$. Then the distances between $\vec{y}^{\,(n)}$ and all clusters in $\mathcal{G}^{(n)}$ (green and turquoise rings as cluster centers) are computed. From the clusters in $\mathcal{G}^{(n)}$, the $C' = 3$ clusters with the smallest distances to $\vec{y}^{\,(n)}$ are selected to define the new $\mathcal{K}^{(n)}$ (which concludes the computations in the second block of Alg. 1). **B** Next iteration starting with the three previously selected clusters, and again showing $\mathcal{K}^{(n)}$ and the search space $\mathcal{G}^{(n)}$ for the new closest clusters. Due to cluster neighborhood overlaps, $|\mathcal{G}^{(n)}|$ is here (and for the following iterations) smaller than the maximum of $C'G$. **C** Third iteration. **D** Fourth and final iteration, any further update does not improve $\mathcal{K}^{(n)}$.

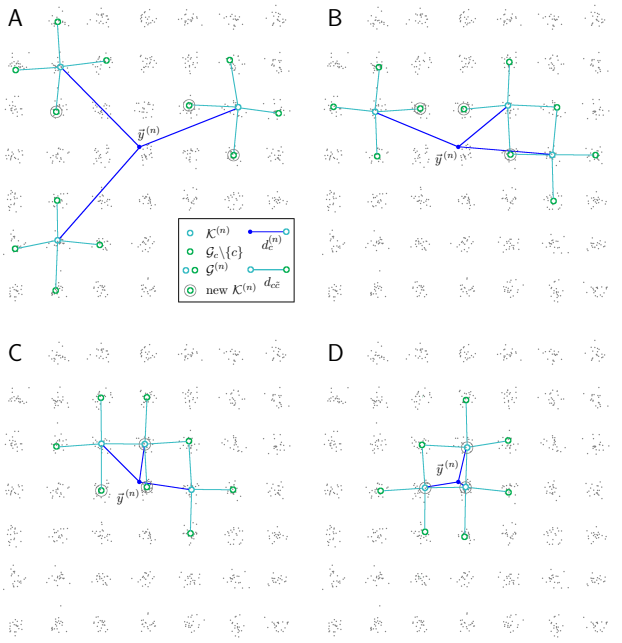

Figure B.1: Visualization of four E-step iterations of Alg. 1. In the fourth iteration the nearest clusters for $\mathcal{K}^{(n)}$ are found.

Note, Alg. 2 finds improved clusters for $\mathcal{K}^{(n)}$ similarly to Alg. 1. The only difference is that the sets $\mathcal{G}_c$ will look less ordered. For Alg. 1 the $\mathcal{G}_c$ in the example of Fig. B.1 form crosses (turquoise lines) because the algorithm always uses the four nearest clusters to construct sets $\mathcal{G}_c$. Instead, Alg. 2 estimates the $\mathcal{G}_c$ which do hence not necessarily contain the nearest neighbors. The $\mathcal{G}_c$ do therefore not look like crosses for this example but also Alg. 2 always decreases the distances (and thus improves the free energy).

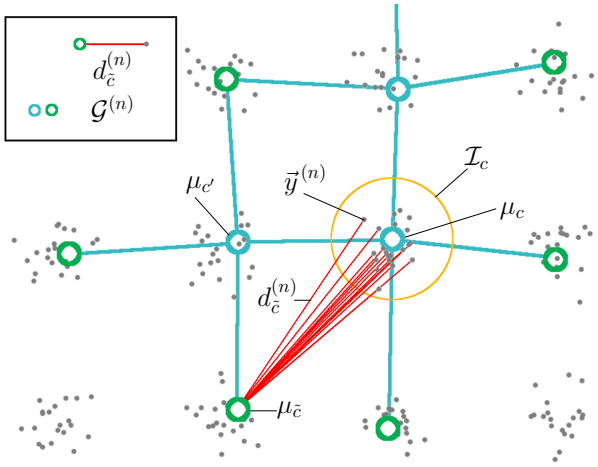

Figure B.2: Visualization of the estimation of cluster-to-cluster distances $d_{c\tilde{c}}$ with already computed cluster-to-point distances $d_{\tilde{c}}^{(n)}$ as used by Alg. 2 in Eq. (8). The mean of the read lines (distances $d_{\tilde{c}}^{(n)}$ of data points closest to cluster c that have cluster $\tilde{c}$ in their own neighborhood search space) gives an estimate of the true distance between cluster c and $\tilde{c}$.

Finally, Fig. B.2 shows an illustration of how the cluster-to-cluster distances $d_{c\tilde{c}}$ are estimated by Alg. 2. For the example, let us consider cluster centers and search spaces per data point $\mathcal{G}^{(n)}$ as in the last iteration of Fig. B.1. In the first computational block, Alg. 2 computes for each data point $\vec{y}^{\,(n)}$ the distances $d_c^{(n)}$ to all clusters in the search space $\mathcal{G}^{(n)}$. From these distances, the algorithm first estimates the set of data points $\mathcal{I}_c$ closest to each cluster. The illustration now shows how the distance $d_{c\tilde{c}}$ between clusters c and $\tilde{c}$ is estimated using Eq. (8). For cluster c first the set of data points $\mathcal{I}_c$ is considered. For most of the points $n \in \mathcal{I}_c$ the distances $d_{\tilde{c}}^{(n)}$ to cluster $\tilde{c}$ have been computed in the first computational block. For instance, data point $\vec{y}^{\,(n)}$ of $\mathcal{I}_c$ (which was used for the illustration in Fig. B.1) has a search space $\mathcal{G}^{(n)}$ which includes cluster $\tilde{c}$ (we illustrate the same $\mathcal{G}^{(n)}$ as for Fig. B.1D). However, data points in the upper-right corner of $\mathcal{I}_c$ have search spaces which do not contain $\tilde{c}$ and are therefore not considered for the summation in Eq. (8). Those data points that are considered for the summation are, however, sufficient to provide a reasonable estimate for the distance $d_{c\tilde{c}}$.

For the example, we have chosen clusters c and $\tilde{c}$ for which the condition in Eq. (8) results in neither trivial nor perfect estimation of $d_{c\tilde{c}}$. If we instead consider clusters c and c' (see Fig. B.2), then distances $d_{c'}^{(n)}$ for all data points of $\mathcal{I}_c$ would be available, and the estimation of $d_{cc'}$ would be very accurate. In general, the closer the clusters or the larger the $\mathcal{G}^{(n)}$ the better are the estimates of cluster-to-cluster distances by Eq. (8).

For clusters very distant to each other, there may be no data points available from which the cluster-to-cluster distance could be estimated. Alg. 2 never estimates these distances, which essentially means that they are considered as infinite. However, in k-means these clusters would likewise never contribute to their respective updates, and for the GMM their contribution would exponentially approach zero with higher distance.

C Further Numerical Results

For Algs. 1 and 2, we monitor the free energy, log-likelihood and standard quantization error during training [see Lücke and Forster, 2017, for the k-means free energy] on a 5×5 BIRCH data set and on KDD. Means and standard errors of the means (SEM), as well as best runs in terms of lowest final quantization error are reported in Fig. C.1 over 100 independent training runs for BIRCH and over 10 runs for KDD (except for standard GMM, where we only show results over a single training run because of its high computational demand).

As already noted, initially (in the first steps) var-GMM and var-k-means require more EM iterations than standard GMM or k-means to obtain comparable quantization errors due to the random $\mathcal{K}^{(n)}/\mathcal{G}^{(n)}$ initialization, which we see here also reflected in the likelihoods. The number of additional EM iterations is however relatively small, and only more significant for very low values of G.

Most importantly for our study, low values of G, which result in strongly decreased run-time complexity, still optimize the clustering objective to values approximately equal to standard GMM and k-means within about the same number of EM iterations (in Fig. C.1 the standard algorithms are $G = 25$ for BIRCH and $G = 200$ for KDD, black lines). For BIRCH, the variational versions were even more effective in avoiding local optima. The best final values of the objective function for var-GMM and var-k-means almost perfectly match those of the best standard k-means and standard GMM runs (black lines), which provides evidence for tight variational likelihood bounds. Our primary goal is comparison to standard GMM and k-means but see, e.g., Lucic et al. [2017] for results on KDD2004 of other recent approaches. As k-means is itself a variational approximation of GMMs [Lücke and Forster, 2017], our partial E-step procedure makes the truncated free energy provably tight. For var-GMM we have finite KL-divergence, but already for relatively small G the free energy becomes almost tight (as the experiments confirm). Also for the large-scale and less regular KDD2004 clustering benchmark, var-GMM and var-k-means match the performance of standard GMM and k-means already for low G ($G = 5$ for var-GMM-S

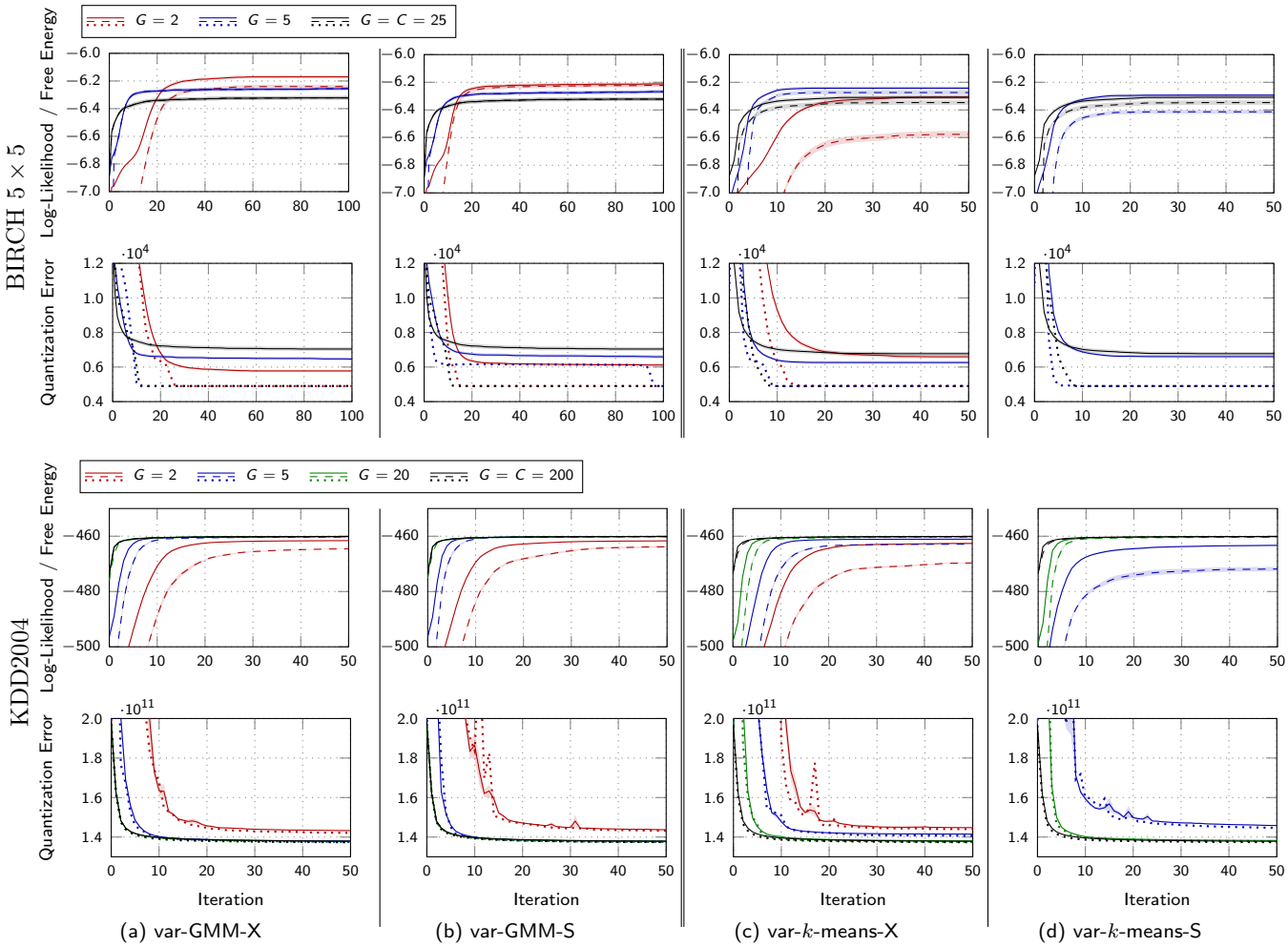

Figure C.1: Results of Algs. 1 and 2 on the BIRCH 5×5 and KDD2004 data sets. Each column (a) - (d) shows results for one specific algorithm for different G-values, depending on the data set. For each data set, the first row shows the mean log-likelihood (solid) and mean free energy (dashed), shaded with their respective SEM. The second row for each data set shows the mean quantization error (solid), shaded with its SEM, as well as the single run with lowest final quantization error (dotted). For var-GMM on KDD the green ($G = 20$) and black ($G = 200$) plots are nearly indistinguishable from another.

and $G = 20$ for var-k-means-S). Lower values of G are still possible, but improved run-times trade off with a decrease in final objective function values.

Var-GMM-S on BIRCH with $G = 2$ needs around ten more iterations to reach similar values than standard GMM with $C = 25$, but each standard GMM iteration requires at least more than six times as many distance computations, making the few additional iterations negligible. For the much larger KDD data set and $C = 200$ clusters, the run-time difference further increase: var-GMM-S requires with $G = 5$ roughly an additional three iterations (i.e., plus three) than standard GMM, but each GMM iteration requires at least eight times the number of data-to-cluster distance evaluations. Still larger are the differences for var-k-means. On BIRCH 5×5, var-k-means-S with $G = 5$ converges basically as fast as k-means and standard GMM (maybe it needs one or two additional iterations), and it optimizes the objective to better final values on average. The speedup per iteration is here at least a factor ($C/G = 5$). On KDD-Cup 2004, var-k-means-S with $G = 20$ optimizes the objective to the same values as k-means and GMM (requiring maybe two to four additional EM iterations). But each var-k-means-S iteration is at least ten times more efficient than standard GMM or k-means. With lower values of G we can obtain even more significant speedups but then the final average objectives are more likely to be worse than for standard GMM and k-means. In general, the more clusters we seek to find in the data, the more significant is the speedup of the variational algorithms (compare Tab. 2).

D Line-by-line Complexity, Algs. 1 & 2

The clustering algorithms for arbitrary size C' of $\mathcal{K}^{(n)}$ and arbitrary size G of $\mathcal{G}_c$ are given by Algs. 1 and 2. In Algs. 3 and 4 we rewrite the same algorithms such that the analysis of the complexity becomes particularly straight-forward. For instance, we duplicate some loops that compute averages to show that such averaging does not increase the complexity. The memory demand for storing all model parameters $\vec{\mu}_c$ and σ, and the new variational parameters $\mathcal{K}^{(n)}$ and nearest neighbors $\mathcal{G}_c$is of $\mathcal{O}(CD + NC' + CG)$. In Alg. 4, also the computed distances $d_c^{(n)}$ have to be stored for all N within each EM iteration (but do not have to be memorized across EM steps), leading to a memory demand of $\mathcal{O}(CD + NC'G+CG)$. For Alg.3, memorizing all the distances is not necessary, as the loop over N of the two last blocks can be combined and the updates can be computed without having to store all distances simultaneously. Similar combinations may be possible for Alg. 4 at least approximately but will require further investigations.

Algorithm 3: Explicit reformulation of Alg. 1.

init $\vec{\mu}_{1:C}$, σ and $\mathcal{K}^{(n)}$ for all n;

repeat

- **for** $c = 1 : C$ **do** — $\boldsymbol{\mathcal{O}(C^2D)}$
  - **for** $\tilde{c} = 1 : C$ **do** — $\mathcal{O}(CD)$
    - $d_{c\tilde{c}} = \|\vec{\mu}_{\tilde{c}} - \vec{\mu}_c\|$; — $\mathcal{O}(D)$
  - $\mathcal{G}_c = \{\tilde{c} \,|\, d_{c\tilde{c}}$ is among the G smallest distances $d_{c:}\}$; — $\mathcal{O}(C)$
- **for** $n = 1 : N$ **do** — $\boldsymbol{\mathcal{O}(NC'GD)}$
  - $\mathcal{G}^{(n)} = \bigcup_{c\in\mathcal{K}^{(n)}} \mathcal{G}_c$; — $\mathcal{O}(C'G)$
  - **for** $c \in \mathcal{G}^{(n)}$ **do** — $\mathcal{O}(C'GD)$
    - $d_c^{(n)} = \|\vec{y}^{\,(n)} - \vec{\mu}_c\|$; — $\mathcal{O}(D)$
  - $\mathcal{K}^{(n)} = \{c \,|\, d_c^{(n)}$ is among the C' smallest distances$\}$; — $\mathcal{O}(C'G)$
- **for** $n = 1 : N$ **do** — $\boldsymbol{\mathcal{O}(NC')}$
  - $\overline{s} = 0$; — $\mathcal{O}(1)$
  - **for** $c \in \mathcal{K}^{(n)}$ **do** — $\mathcal{O}(C')$
    - $s_c^{(n)} = \exp\big(-\frac{1}{2}(d_c^{(n)}/\sigma)^2\big)$; — $\mathcal{O}(1)$
    - $\overline{s} = \overline{s} + s_c^{(n)}$; — $\mathcal{O}(1)$
  - **for** $c \in \mathcal{K}^{(n)}$ **do** — $\mathcal{O}(C')$
    - $s_c^{(n)} = s_c^{(n)} \,/\, \overline{s}$; — $\mathcal{O}(1)$
- update $\vec{\mu}_{1:C}$ and σ^2 using Eqs. (4) with Eq. (5); — $\boldsymbol{\mathcal{O}(NC'D)}$

until $\vec{\mu}_{1:C}$ *and* σ^2 *have converged**;

In principle we could drop the last term $\mathcal{O}(CG)$ in the memory demand of var-GMM-S as $NC'G > CG$. For our purposes, we maintained the last term in order to make the linear C dependence of the memory explicit.

Algorithm 4: Explicit reformulation of Alg. 2.

init $\vec{\mu}_{1:C}$ and σ^2; init $\mathcal{G}^{(n)}$ for all n;

repeat

- **for** $n = 1 : N$ **do** — $\boldsymbol{\mathcal{O}(NC'GD)}$
  - $\mathcal{G}^{(n)} = \bigcup_{c\in\mathcal{K}^{(n)}} \mathcal{G}_c$; — $\mathcal{O}(C'G)$
  - **for** $c \in \mathcal{G}^{(n)}$ **do** — $\mathcal{O}(C'GD)$
    - $d_c^{(n)} = \|\vec{y}^{\,(n)} - \vec{\mu}_c\|$; — $\mathcal{O}(D)$
  - $\mathcal{K}^{(n)} = \{c \,|\, d_c^{(n)}$ is among the C' smallest distances$\}$; — $\mathcal{O}(C'G)$
- **for** $n = 1 : N$ **do** — $\boldsymbol{\mathcal{O}(NC'G)}$
  - $c_o^{(n)} = \operatorname{argmin}_{c\in\mathcal{G}^{(n)}} \{d_c^{(n)}\}$; — $\mathcal{O}(C'G)$
  - $\mathcal{I}_{c_o^{(n)}} = \mathcal{I}_{c_o^{(n)}} \cup \{n\}$; — $\mathcal{O}(1)$
- **for** $c = 1 : C$ **do** — $\boldsymbol{\mathcal{O}(NC'G)}$
  - **for** $n \in \mathcal{I}_c$ **do** — $\mathcal{O}((N/C)C'G)$
    - **for** $\tilde{c} \in \mathcal{G}^{(n)}$ **do** — $\mathcal{O}(C'G)$
      - $d_{c\tilde{c}} = d_{c\tilde{c}} + d_{\tilde{c}}^{(n)}$; — $\mathcal{O}(1)$
      - $b_{c\tilde{c}} = b_{c\tilde{c}} + 1$; — $\mathcal{O}(1)$
- **for** $c = 1 : C$ **do** — $\boldsymbol{\mathcal{O}(NC'G)}$
  - **for** $n \in \mathcal{I}_c$ **do** — $\mathcal{O}((N/C)C'G)$
    - **for** $\tilde{c} \in \mathcal{G}^{(n)}$ **do** — $\mathcal{O}(C'G)$
      - **if** normalized$_{c\tilde{c}} \neq 1$ **then**
        - $d_{c\tilde{c}} = d_{c\tilde{c}}/b_{c\tilde{c}}$; — $\mathcal{O}(1)$
        - normalized$_{c\tilde{c}} = 1$; — $\mathcal{O}(1)$
  - $d_{cc} = 0$; — $\mathcal{O}(1)$
  - $\mathcal{G}_c = \{\tilde{c} \,|\, d_{c\tilde{c}}$ is among the G smallest distances $d_{c:}\}$; — $\mathcal{O}((N/C)C'G)$
- **for** $n = 1 : N$ **do** — $\boldsymbol{\mathcal{O}(NC')}$
  - $\overline{s} = 0$; — $\mathcal{O}(1)$
  - **for** $c \in \mathcal{K}^{(n)}$ **do** — $\mathcal{O}(C')$
    - $s_c^{(n)} = \exp\big(-\frac{1}{2}(d_c^{(n)}/\sigma)^2\big)$; — $\mathcal{O}(1)$
    - $\overline{s} = \overline{s} + s_c^{(n)}$ — $\mathcal{O}(1)$
  - **for** $c \in \mathcal{K}^{(n)}$ **do** — $\mathcal{O}(C')$
    - $s_c^{(n)} = s_c^{(n)} \,/\, \overline{s}$; — $\mathcal{O}(1)$
- update $\vec{\mu}_{1:C}$ and σ^2 using Eqs. (4) with Eq. (5); — $\boldsymbol{\mathcal{O}(NC'D)}$

until $\vec{\mu}_{1:C}$ *and* σ^2 *have converged**;

*Except for $\mathcal{K}^{(n)}$ and $\mathcal{G}_c$, all sets and variables are reset after each EM iteration. First iteration of Alg. 4 uses initial $\mathcal{G}^{(n)}$.