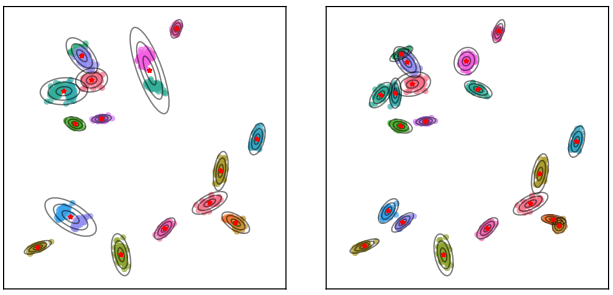
Reinforced EM Algorithm for Clustering with Gaussian Mixture Models

Joshua Tobin* Chin Pang Ho† Mimi Zhang*

Abstract

Methods that employ the EM algorithm for parameter estimation typically face a notorious yet unsolved problem that the initialization input significantly impacts the algorithm output. We here develop a Reinforced Expectation Maximization (REM) algorithm for cluster analysis using Gaussian mixture models. The competence of REM is achieved by introducing two innovative strategies into the EM framework: (1) a mode-finding strategy for initialization that detects non-trivial modes in the data, and (2) a mode-pruning strategy for detecting true modes/mixture components of the population. The pruning strategy is well-justified in the context of mixture modelling, and we present theoretical guarantees on the quality of the initialization. Extensive experimental studies on both synthetic and real datasets show that our approach achieves better performance compared to state-of-the-art methods.



(a) *mclust* [27] (b) *REM*

Figure 1: Comparison between mclust and REM, on a synthetic dataset containing 20 clusters. Optimal clusterings were selected using BIC [24].

1 Introduction

Model-based clustering methods utilize mixture models to partition a collection of objects [12, 4]. The most widely used mixture model is the Gaussian mixture model (GMM). The EM algorithm is the most popular algorithm for parameter estimation of mixture models and known for its sensitivity to initialization [7, Chapter 9]. A multitude of methods for initializing the EM algorithm have been proposed in the literature [6, 16, 20]. A state-of-the-art approach, *mclust* [27], provides hard partitions of the data using a model-based hierarchical clustering algorithm. The desired clusters are obtained from a large number of small clusters through recursive merging [3]. Most initialization methods, including prominent methods that randomly select initial parameters from the data, have the drawback that, when the true number of clusters is not specified, they struggle to provide similar initializations for multiple runs with different cluster numbers, making model selection more difficult. This leads to the key driving force of the present work: to develop a clustering algorithm for GMMs that provides high-quality initializations and produces clusterings that are stable across different cluster numbers.

Our initialization strategy is built on the observation that the modes of a GMM are symptomatic of the underlying population structure and can guide the initialization procedure. In particular, if the Gaussian components in a GMM are well separated, then the modes exactly match the Gaussian means; if the components overlap somewhat, while in some cases, the modes may not be limited to the Gaussian means, they well represent the components of the GMM [2]. Further, the modes of Gaussian mixture densities have been successfully applied for clustering [8, 26]. The peak-finding method [23] returns the instances of the data that best approximate the modes of the density, henceforth referred to as exemplars. Therefore, we apply the peak-finding technique to detect the exemplars in the data. By restricting the Gaussian means to the set of exemplars, the initialization is robust to outliers and adjusts to the location of the centers.

From an inclusive pool of initial exemplars, we produce a hierarchy of clusterings by iteratively pruning superfluous clusters through the optimization of a convex objective function regularized by an adaptive cardinality penalty. We prune exemplars one at a time, automatically generating a nested sequence of clustering results. The optimal clustering is determined by a model selection criterion of the user's choice.

The initialization-pruning framework is called the

*School of Computer Science & Statistics, Trinity College Dublin. {TOBINJO, MIMI.ZHANG}@TCD.IE

†School of Data Science, City University of Hong Kong.

reinforced EM (REM) algorithm. By selecting the Gaussian means from the exemplar set only, the REM algorithm never allows components collapse into one point at which the likelihood is infinite. Furthermore, the objective function for exemplar pruning is well-justified in the context of mixture modelling; it is solved analytically, leading to efficient run time even for large datasets. A comparison of the REM method to *mclust* on a synthetic dataset is presented in Fig. 1.

The remainder of the paper is organized as follows: in Section 2, we describe the EM algorithm for GMMs and review existing approaches; in Section 3, we explain the peak-finding technique; in Section 4, the REM algorithm is described; Section 5 presents experimental evaluations, and Section 6 concludes.

2 Background

Let $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ denote the data matrix: $\boldsymbol{X}^T = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, where the superscript T is the transpose operator. A GMM density has the form $f(\boldsymbol{x}) = \sum_{j=1}^{m} \pi_j \phi(\boldsymbol{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, with mixing proportions π_j ($\pi_j > 0$ and $\sum_{j=1}^{m} \pi_j = 1$), and each Gaussian density $\phi(\cdot; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ has a mean $\boldsymbol{\mu}_j$ and a covariance matrix $\boldsymbol{\Sigma}_j$. Let $\boldsymbol{\pi}$ denote the vector of mixing proportions: $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)^T$. The log-likelihood function is

$$\ell(\boldsymbol{\pi}, \{\boldsymbol{\mu}_j\}_{j=1}^m, \{\boldsymbol{\Sigma}_j\}_{j=1}^m; \boldsymbol{X}) = \sum_{i=1}^{n} \log(\sum_{j=1}^{m} \pi_j \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)).$$

The classical method for computing maximum-likelihood estimates for GMM parameters is the EM algorithm, consisting of the following steps:

1. Initialize the parameters: $\{\pi_1, \ldots, \pi_m\}$, $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m\}$ and $\{\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_m\}$.
2. Compute the responsibilities: for $1 \le i \le n$ and $1 \le j \le m$, $r_{ij} = \frac{\pi_j \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{v=1}^{m} \pi_v \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_v, \boldsymbol{\Sigma}_v)}$.
3. Update the estimates: $\pi_j = \frac{1}{n}\sum_{i=1}^{n} r_{ij}$, $\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{n} r_{ij}\boldsymbol{x}_i}{\sum_{i=1}^{n} r_{ij}}$, and $\boldsymbol{\Sigma}_j = \frac{\sum_{i=1}^{n} r_{ij}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T}{\sum_{i=1}^{n} r_{ij}}$, for $1 \le j \le m$.
4. Iterate steps 2 and 3 until convergence.

The hill-climbing nature of the EM algorithm, coupled with the multi-modal surface of the log-likelihood function, lends crucial importance to the quality of the initialization. The simplest initialization strategy draws initial values at random from the parameter space or from the data pool. Jin et al. [15] proved that, with high probability, the EM algorithm with random initialization will converge to bad local maxima, whose log-likelihood could be arbitrarily worse than that of the global maximum. Another intuitive idea is to run EM with multiple random starts for each value of m. The *emEM* algorithm [6] consists of several short runs of EM, initialized with random starts, until a loose convergence criterion is satisfied. The solution with the highest log-likelihood is used to initialize a long run of EM with strict converge criteria. A related approach, called *Rnd-EM* [16], computes the log-likelihood of several random starts without running any EM iterations. The best is used as the initializer for the long EM stage. The k-means algorithm is also frequently used to provide an initial partition of the data, from which initial parameter estimates can be computed. However, the k-means algorithm itself requires a good initialization, typically achieved with the k-means++ method [28]. None of these methods make efforts to ensure the similarity of initializations for different cluster numbers. This leads to unstable clusterings and hinders comparison of clusterings using model selection criteria.

The popular *mclust* package in R provides hard partitions of the data using an agglommerative model-based clustering technique. Initializations for different numbers of clusters are found from partitions extracted using dendrogram derived from the hierarchical clustering. This approach provides a hard partition of the data, leading to improper splitting of true components when the estimated cluster number is greater than the true number of components.

To decide the optimal cluster number via a model selection criterion, three criteria predominate in the literature: the Akaike information criterion (AIC) [1], the Bayesian information criterion (BIC) [24], and the integrated completed likelihood (ICL) [5].

3 Exemplar Selection

The peak-finding technique is adopted to detect the initial set of exemplars [23]. It requires two inputs: (1) a density estimate at each data point, and (2) the distance from each point to its nearest neighbor of higher density. We apply the Gaussian kernel with bandwidth $h > 0$ for density estimation:

$$f_h(\boldsymbol{x}) = \frac{1}{n \cdot h^p} \sum_{i=1}^{n} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right),$$

where $K(\cdot)$ is the Gaussian kernel. Note that our density estimate is different from that in [23]; it is more smooth and better suits the data that are generated from GMMs. For the distance input, define

$$b(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}' \in \boldsymbol{X}} \{\|\boldsymbol{x} - \boldsymbol{x}'\| : f_h(\boldsymbol{x}) < f_h(\boldsymbol{x}')\},$$

i.e. the nearest neighbor of $\boldsymbol{x}$ with a higher density. Then the distance from $\boldsymbol{x}$ to its nearest neigh-

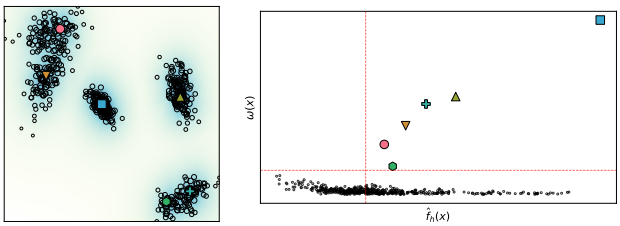


Figure 2: An example with 5 clusters. The left plot shows the estimated density of the data, with darker regions having higher density. Also shown are the locations of the exemplars, with colour and marker type corresponding to the instances in the right plot. The right plot is the decision plot, where the red lines indicate the threshold values. The peak-finding method selects six initial exemplars, with each true component well represented.

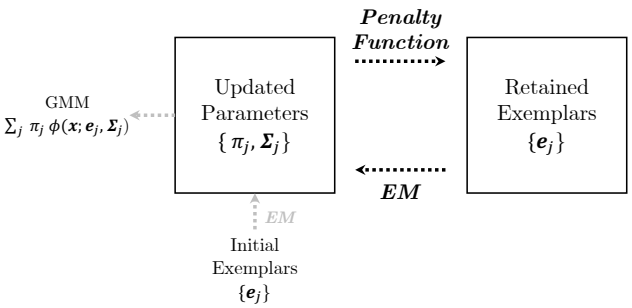


Figure 3: Given any exemplar set $\{\boldsymbol{e}_j\}$, we estimate the parameters $\{\pi_j, \boldsymbol{\Sigma}_j\}$ by an EM-type algorithm. At convergence, we obtain a GMM $\sum_j \pi_j \phi(\boldsymbol{x}; \boldsymbol{e}_j, \Sigma_j)$. Then we optimize a regularized objective function to force certain π_j's to be 0. The relevant exemplars are removed from the exemplar set. The procedure is repeated producing a sequence of clusterings.

bor of higher density is simply $\omega(\boldsymbol{x}) = \|\boldsymbol{x} - b(\boldsymbol{x})\|$. For the point with the highest density estimate $\boldsymbol{x} = \arg\max_{\boldsymbol{x}' \in \boldsymbol{X}} f_h(\boldsymbol{x}')$, the distance is defined as $\omega(\boldsymbol{x}) = \max_{\boldsymbol{x}' \in \boldsymbol{X}} \|\boldsymbol{x} - \boldsymbol{x}'\|$.

Intuitively, $\omega(\boldsymbol{x})$ will be large if $\boldsymbol{x}$ has a locally or globally maximal density, or if $\boldsymbol{x}$ is an outlier. Therefore, a data point $\boldsymbol{x}$ will be selected as an exemplar by the peak-finding method, only if both $f_h(\boldsymbol{x})$ and $\omega(\boldsymbol{x})$ are large. To generate an initial set of exemplars $\mathcal{E}_0 = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_\kappa\}$, threshold values for the density $f_h(\boldsymbol{x})$ and the distance $\omega(\boldsymbol{x})$ need to be set: the exemplars are the data points with the two metric values both above the thresholds, i.e. $\mathcal{E}_0 = \{\boldsymbol{x} \in \boldsymbol{X} : f_h(\boldsymbol{x}) \geq l, \omega(\boldsymbol{x}) \geq \tau\}$. REM provides users with a decision graph, a scatter plot of $\{(f_h(\boldsymbol{x}), \omega(\boldsymbol{x})) : \boldsymbol{x} \in \boldsymbol{X}\}$, to provide intuition regarding the threshold values. An example of the decision graph and the selected exemplars is provided in Figure 2.

We provide theoretical guarantees on mode recovery in the supplementary material. The theoretical results claim that the peak-finding technique recovers the modes of a GMM density consistently. For n large enough, with high probability, $\mathcal{E}_0$ contains unique estimates for all the true modes of the GMM. For the clustering task, the instances selected by the peak-finding criterion provide appropriate estimates for the mean parameters of the Gaussian components. For well-separated components, the modes of the GMM and the mean parameters correspond exactly. In the case of overlapping components, the inclusive set of mode estimates produced by the peak-finding technique model successfully models the high-density regions of the GMM. Through the iterative pruning and parameter estimation steps to be introduced in Section 4, the initial set of exemplars is reduced to contain only those instances which well-represent their associated Gaussian components.

4 The Iterative Pruning Procedure

The REM algorithm has two blocks: the EM block and the pruning block; see Figure 3. Given the initial exemplar set $\mathcal{E}_0 = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_\kappa\}$, our iterative procedure will produce a sequence of nested clustering results, respectively with $\kappa - 1, \kappa - 2, \ldots, 2$ mixture components. An example is given in the supplementary material. The final optimal clustering is determined by a model selection criterion of the user's choice. Note that, if a data point is in the exemplar set, it is excluded from the data pool, and once a data point is pruned from the exemplar set, it will go back to the data pool. In other words, if $\boldsymbol{e}_j$ is pruned in the pruning block, then we update $\mathcal{E} = \mathcal{E} \backslash \{\boldsymbol{e}_j\}$, $\boldsymbol{X} = \boldsymbol{X} \cup \{\boldsymbol{e}_j\}$ and $n = n + 1$, before running the EM block.

4.1 EM Block Given the updated data pool (the original data without the retained exemplars), the EM block operates as follows.

1. Input: The retained exemplars $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_\kappa\}$ and the responsibilities $\{r_{ij} : i = 1, \ldots, n, j = 1, \ldots, \kappa\}$.

2. Update the estimates: for $j = 1, \ldots, \kappa$,

$$\pi_j = \frac{\sum_{i=1}^n r_{ij}}{n}, \boldsymbol{\Sigma}_j = \frac{\sum_{i=1}^n r_{ij}(\boldsymbol{x}_i - \boldsymbol{e}_j)(\boldsymbol{x}_i - \boldsymbol{e}_j)^T}{\sum_{i=1}^n r_{ij}}.$$

3. Compute the responsibilities: for $i = 1, \ldots, n$ and

$j = 1, \ldots, \kappa$,

$$r_{ij} = \frac{\pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)}{\sum_{v=1}^{\kappa} \pi_v \phi(\boldsymbol{x}_i; \boldsymbol{e}_v, \boldsymbol{\Sigma}_v)}.$$

4. Iterate steps 2 and 3 until convergence.

In the EM block, the Gaussian means are fixed at the given exemplars, and only the mixing proportions and covariance matrices are estimated. Since the exemplars are excluded from the data pool, the iteration will never converge to a degenerate solution. The GMM density at convergence is

$$f(\boldsymbol{x}) = \sum_{j=1}^{\kappa} \pi_j \phi(\boldsymbol{x}; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j).$$

4.2 Pruning Block While the original exemplar set contains consistent estimates of all the density peaks of the mixture model, the number of density peaks can be significantly larger than the number of mixture components [2]. Hence, the exemplars $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_\kappa\}$ need be further filtered to obtain the true mean vectors.

In the pruning block, we prune exemplars by inducing sparsity in the mixing proportion vector $\boldsymbol{\pi}$ such that, if $\pi_j = 0$, then the exemplar $\boldsymbol{e}_j$ will be removed from the exemplar set and returned to the data pool. Let $\mathbf{1}_n$ denote the vector of 1's of dimension n; let Δ denote the probabilistic simplex of the appropriate dimension: if $\boldsymbol{\pi} \in \Delta$, then $\boldsymbol{\pi} \geq 0$ and $\|\boldsymbol{\pi}\|_1 = 1$. Given the exemplar set $\mathcal{E}_0$ and covariance-matrix estimates $\{\boldsymbol{\Sigma}_j\}_{j=1}^{\kappa}$, let $\boldsymbol{D} = [d_{ij}]_{n \times \kappa}$ denote the squared-distance matrix, where $d_{ij} = (\boldsymbol{x}_i - \boldsymbol{e}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{e}_j)$.

4.2.1 The Objective Function Given the exemplars, the log-likelihood function is $\sum_{i=1}^{n} \log\left(\sum_{j=1}^{\kappa} \pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)\right)$. We have by Jensen's inequality that

$$-\log\left(\sum_{j=1}^{\kappa} \pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)\right) = \left(\sum_{j=1}^{\kappa} r_{ij}\right) \log\left(\frac{\sum_{j=1}^{\kappa} r_{ij}}{\sum_{j=1}^{\kappa} \pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)}\right) \leq \sum_{j=1}^{\kappa} r_{ij} \log\left(\frac{r_{ij}}{\pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)}\right),$$

where the upper bound is achievable following the log-sum inequality. Then the negative log-likelihood can be formulated as an optimization problem:

$$-\sum_{i=1}^{n} \log\left(\sum_{j=1}^{\kappa} \pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)\right) = \min_{\{\boldsymbol{R}_{i\cdot} \in \Delta\}_{i=1}^{n}} \sum_{i=1}^{n} \sum_{j=1}^{\kappa} r_{ij} \log\left(\frac{r_{ij}}{\pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)}\right),$$

where $\boldsymbol{R} = [r_{ij}]_{n \times \kappa}$ and $i\cdot$ indicates the ith row. Then maximizing the log-likelihood is equivalent to minimizing $\sum_{i=1}^{n} \sum_{j=1}^{\kappa} r_{ij} \log\left(\frac{r_{ij}}{\pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)}\right)$. In the pruning block, the $\boldsymbol{e}_j$'s and $\boldsymbol{\Sigma}_j$'s are fixed, and hence the optimization variables are the responsibilities only:

$$\min_{\{\boldsymbol{R}_{i\cdot} \in \Delta\}_{i=1}^{n}} \sum_{i=1}^{n} \sum_{j=1}^{\kappa} r_{ij} \log\left(\frac{r_{ij}}{\pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)}\right).$$

The detailed formulation of the objective function is

$$\sum_{i=1}^{n} \sum_{j=1}^{\kappa} r_{ij} \log\left(\frac{r_{ij}}{\pi_j \phi(\boldsymbol{x}_i; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j)}\right) = \sum_{i=1}^{n} \sum_{j=1}^{\kappa} r_{ij} \times \left[\log\left(\frac{r_{ij}}{\pi_j}\right) + \frac{1}{2} \log\left(|\boldsymbol{\Sigma}_j|\right) + \frac{1}{2} (\boldsymbol{x}_i - \boldsymbol{e}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{e}_j)\right].$$

To penalize $\boldsymbol{\pi}$, we take out the first term in the brackets; otherwise, numerical algorithms will behave erratically when $\pi_j \to 0$. Further motivation for the removal of this term is given in the supplementary material.

This yields the optimization problem

$$\min_{\{\boldsymbol{R}_{i\cdot} \in \Delta\}_{i=1}^{n}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{\kappa} r_{ij} (\boldsymbol{x}_i - \boldsymbol{e}_j)^T \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{x}_i - \boldsymbol{e}_j) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{\kappa} r_{ij} \log\left(|\boldsymbol{\Sigma}_j|\right).$$

In matrix-vector notation, the optimization problem is

$$\min_{\{\boldsymbol{R}_{i\cdot} \in \Delta\}_{i=1}^{n}} \frac{1}{2} \sum_{i=1}^{n} \boldsymbol{R}_{i\cdot}^T (\boldsymbol{D}_{i\cdot} + \boldsymbol{\xi}), \tag{4.1}$$

where $\boldsymbol{\xi} = (\log(|\boldsymbol{\Sigma}_1|), \ldots, \log(|\boldsymbol{\Sigma}_\kappa|))^T$.

4.2.2 The Penalty We apply the classical ℓ_1-norm penalty on $\boldsymbol{\delta} \circ \boldsymbol{\pi}$: $\|\boldsymbol{\delta} \circ \boldsymbol{\pi}\|_1 = \boldsymbol{\delta}^T \boldsymbol{\pi}$, where $\circ$ is the element-wise multiplication operator, and $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_\kappa)^T$ is a weight vector. We here define δ_i as the probability that an instance from the ith mixture component is misclassified into the jth mixture component:

$$\delta_i = \max_{j=1,\ldots,\kappa} \Pr\left(\pi_i \phi(\boldsymbol{x}; \boldsymbol{e}_i, \boldsymbol{\Sigma}_i) < \pi_j \phi(\boldsymbol{x}; \boldsymbol{e}_j, \boldsymbol{\Sigma}_j) | \boldsymbol{x} \sim N(\boldsymbol{e}_i, \boldsymbol{\Sigma}_i)\right).$$

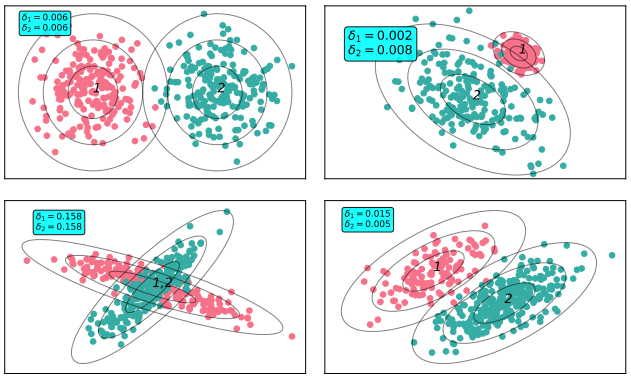


Figure 4: The overlap penalty for typical scenarios with two components: (1) equal weights and homogeneous spherical covariance matrices, (2) equal weights and heterogeneous covariance matrices, (3) equal weights, common mean and heterogeneous covariance matrices, (4) unequal weights and homogeneous covariance matrices.

This definition was introduced by Maitra and Melnykov [17] to measure the degree of overlap between two Gaussian distributions. The weight δ_i reflects the likelihood of the exemplar $\boldsymbol{e}_i$ belonging to the group of another exemplar. Exemplars favoured by this penalty are in keeping with the peak-finding conception of cluster centers as instances with high density, and relatively large distance to other points of higher density.

For Gaussian distributions with homogeneous covariance matrices, the computation of the probability is straightforward and related to the Mahalanobis distance between the exemplars. For general covariance matrices, the computation involves evaluating the cumulative distribution function of linear combinations of independent non-central chi-squared and normal random variables. A method using the algorithm AS 155 of [10] has been implemented in C as part of the *MixSim* package in R [19]. Example values of the δ_i's for four typical scenarios of two-component mixtures are provided in Figure 4.

4.2.3 Objective Minimization Our penalized objective function is

$$\min_{\{\boldsymbol{R}_{i\cdot}\in\Delta\}_{i=1}^n} \quad \frac{1}{2}\sum_{i=1}^n \boldsymbol{R}_{i\cdot}^T(\boldsymbol{D}_{i\cdot}+\boldsymbol{\xi}) + \theta\boldsymbol{\delta}^T\boldsymbol{R}^T\mathbf{1}_n, \tag{4.2}$$

where the regularization term will force certain columns of $\boldsymbol{R}$ to be exactly zero. The objective function (4.2) is linear and hence can be simplified to be

$$\min_{\{\boldsymbol{R}_{i\cdot}\in\Delta\}_{i=1}^n} \sum_{i=1}^n \boldsymbol{R}_{i\cdot}^T\boldsymbol{b}_i = \sum_{i=1}^n \min_{\boldsymbol{R}_{i\cdot}\in\Delta} \boldsymbol{R}_{i\cdot}^T\boldsymbol{b}_i,$$

where $\boldsymbol{b}_i = \frac{1}{2}\boldsymbol{D}_{i\cdot} + \frac{1}{2}\boldsymbol{\xi} + \theta\boldsymbol{\delta}$. That is, our optimization problem can be decomposed into n independent optimization problems. Moreover, the solution to each individual problem $\boldsymbol{R}_{i\cdot}^* = \min_{\boldsymbol{R}_{i\cdot}\in\Delta} \boldsymbol{R}_{i\cdot}^T\boldsymbol{b}_i$ is trivial:

$$r_{ij}^* = \begin{cases} 1, & \text{if } j = \arg\min_{1\le v\le\kappa}\{b_v\}; \\ 0, & \text{otherwise.} \end{cases}$$

The parameter θ controls the amount of shrinkage on $\boldsymbol{\pi}(=\frac{1}{n}\boldsymbol{R}^T\mathbf{1}_n)$. The solution path is piecewise linear in θ, and the whole trajectory of $\boldsymbol{\pi}$, as a function of θ, can be easily computed by piecewise-linear homotopy methods (details given in the supplementary material). Given the solution $\boldsymbol{R}^*$ to problem (4.2) with one zero-column, the refined exemplar set is $\mathcal{E}_1 = \{\boldsymbol{e}_j \in \mathcal{E}_0 : \pi_j^* \neq 0\}$. An example plotting the trajectory of $\boldsymbol{\pi}$ in each iteration is also given in the supplementary material.

5 Evaluation

The performance of the REM algorithm is demonstrated on synthetic and real-world datasets. We compare REM with popular and state-of-the-art initialization methods. The following methods are used for comparison:

Random Initialization (riEM) Random sampling from the data pool, and the EM algorithm is run to convergence. Available in Scikit-Learn [22].

***k*-means++ Initialization (kmEM)** The k-means++ algorithm is used to provide initial partitions of the data. Available in Scikit-Learn [22].

emEM [6] Truncated runs of EM with random initializations provide initial parameter values. Implemented as a wrapper for the Scikit-Learn library.

rndEM [16] Similar to emEM with truncated runs lasting only one iteration. Implemented as a wrapper for the Scikit-Learn library.

mclust [27] Model-based hierarchical clustering provides initial partitions of the data. Implemented in R and C as part of the *mclust* package.

REM is implemented in Python and can be readily invoked. The code for REM and for reproducing the below experiments is available online.[1]

We report the execution time for each method to produce the optimal clustering and the number of clusters detected. We adopt the widely used Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) for performance evaluation.

5.1 Experimental Setup The bandwidth value in the Gaussian kernel is set to be the average of the distances from instances to their k-th nearest neighbor, where $k = \min(\sqrt{n}, 30)$. The impact of the bandwidth parameter on REM performance is further discussed in

[1] https://github.com/tobinjo96/REM (Github Repository)

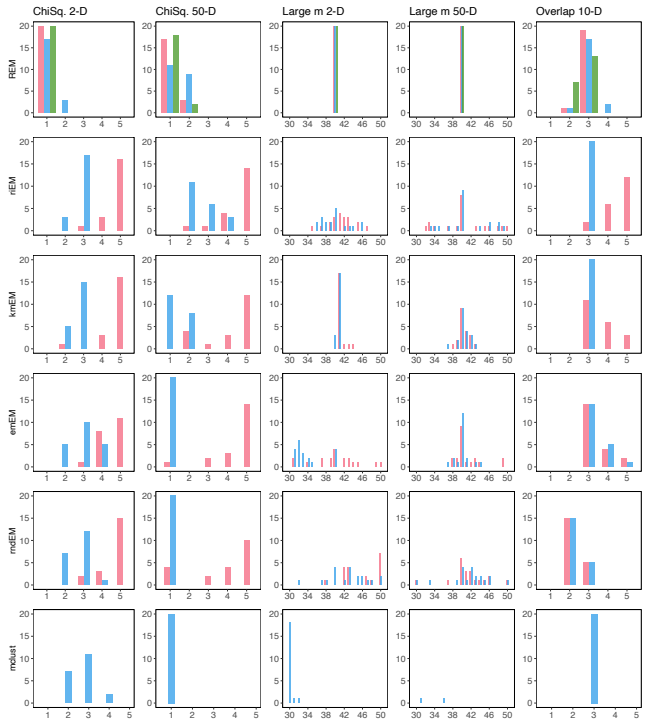


Figure 5: The number of clusters returned by each algorithm using the AIC (pink), BIC (blue), and ICL (green) model selection criteria.

Section 5.4 below. For all the benchmark methods, the range for the number of clusters is required as an input. If κ exemplars are selected for a given dataset by REM, then for competitor methods, the cluster number m will range from 1 to $\kappa + 2$. For each value of m, riEM is initialized 25 times, kmEM is initialized 25 times, rndEM is initialized 200 times, emEM is initialized 50 times (with the maximum number of iterations of the truncated EM set to 50, and the tolerance for convergence of the log-likelihood set to 1×10^{-3}). The model selection criteria for REM include the AIC, BIC, and ICL. For riEM, kmEM, emEM, and rndEM, the AIC and BIC are used. For mclust, the authors note a preference for the BIC. For fair comparison between methods, we allow the covariance matrices to take any form. The tolerance for convergence of the log-likelihood for EM is set to 1×10^{-5}, and the maximum number of iterations is set to 100 for each method. All experiments were conducted on a PC running Debian 10 (Buster), consisting of 24 cores and 24GB of RAM.

5.2 Simulated Datasets We first examine the performance of the peak-finding method when no cluster structure is present in the data. Following [25], we generate 40 datasets, 20 of dimension $p = 2$ and $p = 50$ respectively, from independent χ^2 distributions with 10 degrees of freedom. The competitor methods are initialized with $m \in [1, 5]$. The number of clusters returned by the competitor methods and REM for each of the datasets is shown in Figure 5 (the Chi-squared columns). The peak-finding method is adept at correctly choosing only one potential exemplar in the data. Since the data are skewed, the competitor methods detect more than one component in the data. This shows the efficiency of the peak-finding method for initialization.

We next consider datasets with a large number of well-separated clusters. Again, 40 datasets are generated, 20 each of dimension $p = 2$ and $p = 50$ respectively. The *MixSim* package in R is used to generate 40 clusters with no overlap. The competitor methods were initialized with $m \in [30, 50]$. The number of clusters returned by the competitor methods and REM for each of the datasets is shown in Figure 5 (the Large m columns). The advantages of deterministic initialization methods are clear in this case. Furthermore, kmEM and REM achieve the correct number of components for $p = 2$, while the remaining stochastic initialization methods struggle to place initial mean vectors in every cluster. Due to the high number of parameters in these models, mclust returns a solution for only two of the 50-dimensional datasets. REM achieves the correct number of clusters for each model selection criterion.

Finally, we consider 20 simulations of the mixture provided in Experiment 3 of [21]. This mixture consists of three overlapping clusters in 10 dimensions, two with adjacent mean vectors. The competitor methods are initialized with $m \in [1, 5]$. The number of clusters returned by the competitor methods and REM for each of the datasets is shown in Figure 5 (the Overlap column). In this case, REM achieves the correct number of components for the AIC and BIC model-selection criteria for each dataset. The ICL criterion merges the components with adjacent mean vectors, in keeping with its preference for sparser models relative to the other two criteria. For the competitor methods, mclust achieves the correct number of clusters for each dataset and the BIC is seen to generally outperform the AIC.

Name		$\boldsymbol{n}$	$\boldsymbol{p}$	$\boldsymbol{m}$
Ecoli	[11]	336	7	8
G2	[18]	2048	128	2
Iris	[11]	150	4	3
Satellite	[11]	4435	36	6
Seeds	[11]	210	7	3
Wine	[11]	178	14	3

Table 1: The characteristics of the evaluated datasets.

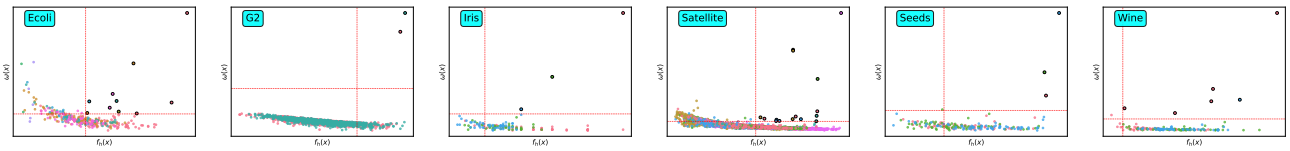


Figure 6: The decision graphs for the six real datasets with color representing the true class label. The dashed lines show the values of τ and l that decide the initial exemplars.

Dataset	Metric	REM AIC	REM BIC	REM ICL	riEM AIC	riEM BIC	kmEM AIC	kmEM BIC	emEM AIC	emEM BIC	rndEM AIC	rndEM BIC	Mclust BIC
Ecoli	ARI	0.599	0.599	0.599	0.600	0.282	**0.619**	0.277	0.608	0.233	0.607	0.294	0.560
	NMI	0.566	0.566	0.566	**0.601**	0.382	0.598	0.370	0.596	0.275	0.574	0.385	0.551
G2	ARI	**1.000**	**1.000**	**1.000**	0.148	0.000	0.699	0.003	0.748	0.000	**1.000**	0.000	0.000
	NMI	**1.000**	**1.000**	**1.000**	0.185	0.004	0.723	0.051	0.764	0.000	**1.000**	0.000	0.000
Iris	ARI	**0.904**	**0.904**	**0.904**	0.742	0.443	0.693	0.693	0.856	0.568	**0.904**	0.568	0.562
	NMI	**0.900**	**0.900**	**0.900**	0.791	0.649	0.769	0.769	0.857	0.734	**0.900**	0.734	0.761
Satellite	ARI	**0.524**	**0.524**	**0.524**	0.443	0.419	0.444	0.465	0.423	0.439	0.465	0.476	0.467
	NMI	**0.578**	**0.578**	**0.578**	0.501	0.490	0.565	0.556	0.553	0.521	0.549	0.556	0.563
Seeds	ARI	0.766	0.766	0.766	0.594	0.663	0.576	0.621	0.624	0.624	0.644	0.663	**0.788**
	NMI	**0.744**	**0.744**	**0.744**	0.643	0.621	0.646	0.663	0.663	0.663	0.669	0.621	**0.744**
Wine	ARI	**0.534**	0.501	0.501	0.480	0.008	0.502	0.510	0.507	0.510	0.476	0.510	0.498
	NMI	0.526	0.597	0.597	0.559	0.072	0.508	0.573	0.584	0.573	0.520	0.601	**0.613**

Table 2: Clustering results on the real datasets by different methods. The best results are highlighted in bold.

5.3 Real Datasets Six datasets are used to compare the clustering performance of REM with the competitor methods. Details of the datasets can be found in Table 1. Instances with missing values were removed. For each dataset, the initial exemplars for the REM algorithm are indicated in the decision graph in Figure 6. For the G2, Iris and Seeds datasets, the number of components is immediately obvious from the decision graph. For the other three datasets, we set the two threshold values to include all promising exemplars.

The results for clustering the six datasets are presented in Table 2. For each method, the ARI and NMI values are calculated from the clustering decided by the relevant model-selection criterion. REM achieves the best clustering, in terms of NMI, for four of the six datasets examined and, in terms of ARI, the best for five of the six datasets. Moreover, the clustering results from REM are consistent across the different model-selection criteria. This is due to the fact that, in REM, the mean vectors are fixed at the exemplars. Mclust, the other deterministic initialization method assessed, outperforms REM for the Seeds dataset, and achieves comparable performance for the Satellite and Wine datasets. Unlike REM, which achieves the perfect clustering on the G2 dataset, Mclust is not able to detect the two clusters, as the BIC criterion merges the two components in search of a sparse model. The stochastic initialization approaches are capable of achieving excellent results, for example the rndEM on the Iris dataset. However, the performance is not consistent between different runs, and the results are not robust to the choice of the model-selection criterion.

Dataset	REM	riEM	kmEM	emEM	rndEM	Mclust
Ecoli	**1.42**	214.7	71.6	19.6	181.9	2.4
G2	**32.0**	2025.9	52.1	181.1	2433.0	1153.9
Iris	**0.3**	90.9	4.7	2.26	26.3	0.7
Satellite	**295.0**	7302.2	1161.9	1020.4	7242.5	437.7
Seeds	**0.9**	158.2	2.3	4.01	41.2	1.2
Wine	3.9	142.8	4.1	4.69	57.6	**0.4**

Table 3: Execution time (seconds) for the evaluated datasets.

The run time, in seconds, for each of the methods is presented in Table 3. For small datasets, we see that REM is competitive with Mclust, and both are much faster than the other methods. REM runs faster for the low-dimensional datasets, whereas Mclust is faster for the Wine dataset. The magnitude of difference is negligible and unlikely to hinder the use of REM in ap-

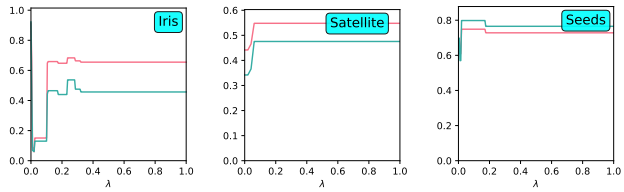


Figure 7: The performance of the REM algorithm evaluated by the ARI (pink) and NMI (blue) as the parameter h changes.

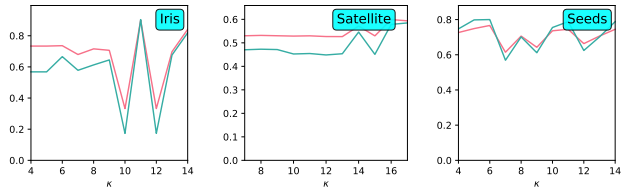


Figure 8: The performance of the REM algorithm evaluated by the ARI (pink) and NMI (blue) as the number of selected exemplars changes.

plications. We see that, for larger datasets, REM has the fastest run time. The difference is most pronounced for the G2 dataset and the Satellite dataset. The execution times of the stochastic methods are significantly higher than Mclust and REM. For riEM and rndEM, this is caused by the slow convergence of the EM algorithm as a result of naive initializations. For emEM and kmEM, providing the initializations requires significant computation, slowing down execution significantly.

5.4 Ablation Study REM has two tuning parameters: (1) h, the bandwidth of the Gaussian kernel and (2) κ, the number of exemplars selected from the decision graph. To examine the impact of h and κ on the REM algorithm, we use, for brevity, three real datasets (Iris, Satellite, Seeds), covering a range of sizes and dimensions, with the remainder are included in the supplementary material.

To assess the impact of the bandwidth parameter, we compute the minimum (d_{min}) and maximum (d_{max}) pairwise distances in the data, and set $h = d_{min} + \lambda(d_{max} - d_{min})$. We increase λ from 0 to 1 in increments of 0.005 and run REM for each increment. We can see from Figure 7 that, for the Iris, Satellite and Seeds datasets, the results are robust to the choice of h, with REM achieving high quality clusterings over a very broad range of λ values.

To assess the impact of κ on the clustering quality, we run REM on the four datasets ten times, initialized with $\kappa \in [m, m+10]$ centers. The selected exemplars are the κ instances with highest values of $f_h(\boldsymbol{x}) \times \omega(\boldsymbol{x})$. The results, shown in Figure 8, demonstrate that the REM algorithm is robust to the number of initial exemplars. This confirms the intuition that the penalty introduced in Section 4.2.2 correctly prunes spurious exemplars.

	Peaks	Prune	Fixed	ARI	NMI
M1		✓		0.237	0.283
M2	✓			0.571	0.541
M3	✓		✓	0.417	0.563
M4	✓	✓		0.578	0.552
REM	✓	✓	✓	**0.599**	**0.566**

Table 4: Clustering results on the Ecoli dataset by the ablation methods.

The importance of the main features of REM are demonstrated by individually substituting (1) the peak-finding exemplar selection with random selection, (2) the iterative pruning algorithm with an ordering of exemplars based on the magnitude of $f_h(\boldsymbol{x}) \times \omega(\boldsymbol{x})$, and (3) the EM-type algorithm with fixed means with the full EM algorithm detailed in Section 3. The results for the Ecoli dataset are shown in Table 5 and the remaining datasets are included in the supplementary material. Random initialization leads to significantly poorer results than the peak-finding method as seen when comparing methods M1 and M2 in the Table 5. The impact of the pruning approach is also clear when comparing methods M2 and M4. Finally, it is noted that REM achieves the best result for each metric on each of the datasets assessed, highlighting the mutually beneficial impact of each of the constituent parts.

6 Conclusion and Future Work

This work introduced REM, an algorithmic tool for model-based clustering, which extricates the EM algorithm from the initialization problem. We showed that the peak-finding method is an effective tool for quickly determining high-quality exemplars. For exemplar pruning, we developed a novel objective function that originates from the log-likelihood function, integrates a data-driven penalty, admits analytic solutions, and allows distributed computing. Through iterative pruning of the exemplars, our algorithm generates a sequence of nested clusterings, from which the preferred partition can be selected. Experimental results demonstrated that our method has excellent performance. It achieves perfect clusterings for datasets containing well-separated clusters and outperforms prominent benchmark methods on a broad range of simulated and real-world datasets. The hyper-parameters of our model are conventional and can be handily tuned by users. We

showed that our algorithm achieves consistent results over a broad range of hyper-parameter values. In future, we envisage incorporating structured covariance matrices into our method to allow for even faster computation.

Acknowledgements

The authors acknowledge funding from the HEA, DFHERIS and the Shared Island Fund.

References

[1] H. Akaike, *A new look at the statistical model identification*, IEEE Transactions on Automatic Control, 19 (1974), pp. 716–723.

[2] C. Améndola, A. Engström, and C. Haase, *Maximum Number of Modes of Gaussian Mixtures*, Information and Inference: A Journal of the IMA, 9 (2020), pp. 587–600.

[3] J. D. Banfield and A. E. Raftery, *Model-Based Gaussian and Non-Gaussian Clustering*, Biometrics, 49 (1993), pp. 803–821.

[4] J.-P. Baudry, A. E. Raftery, G. Celeux, K. Lo, and R. Gottardo, *Combining mixture components for clustering*, Journal of Computational and Graphical Statistics, 19 (2010), pp. 332–353.

[5] C. Biernacki, G. Celeux, and G. Govaert, *Assessing a mixture model for clustering with the integrated completed likelihood*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), pp. 719–725.

[6] C. Biernacki, G. Celeux, and G. Govaert, *Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models*, Computational Statistics & Data Analysis, 41 (2003), pp. 561–575.

[7] C. Bishop, *Pattern Recognition and Machine Learning*, Information Science and Statistics, Springer-Verlag New York, first ed., 2006.

[8] J. E. Chacón, *Mixture model modal clustering*, arXiv:1609.04721 [stat], (2016).

[9] S. Dasgupta and S. Kpotufe, *Optimal rates for k-NN density and mode estimation*, Advances in Neural Information Processing Systems, 27 (2014).

[10] R. B. Davies, *Algorithm AS 155: The Distribution of a Linear Combination of X2 Random Variables*, Journal of the Royal Statistical Society. Series C (Applied Statistics), 29 (1980), pp. 323–333.

[11] D. Dua and C. Graff, *UCI Machine Learning Repository*, 2019.

[12] C. Fraley and A. E. Raftery, *Model-based clustering, discriminant analysis, and density estimation*, Journal of the American Statistical Association, 97 (2002), pp. 611–631.

[13] H. Jiang, *On the Consistency of Quick Shift*, Advances in Neural Information Processing Systems, 30 (2017).

[14] ——, *Uniform Convergence Rates for Kernel Density Estimation*, in Proceedings of the 34th International Conference on Machine Learning, PMLR, July 2017, pp. 1694–1703.

[15] C. Jin, Y. Zhang, S. Balakrishnan, M. J. Wainwright, and M. I. Jordan, *Local maxima in the likelihood of gaussian mixture models: Structural results and algorithmic consequences*, in Advances in Neural Information Processing Systems 29, 2016, pp. 4116–4124.

[16] R. Maitra, *Initializing Partition-Optimization Algorithms*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 6 (2009), pp. 144–157.

[17] R. Maitra and V. Melnykov, *Simulating Data to Study Performance of Finite Mixture Modeling and Clustering Algorithms*, Journal of Computational and Graphical Statistics, 19 (2010), pp. 354–376.

[18] P. F. R. Mariescu-Istodor and C. Zhong, *Xnn graph*, LNCS 10029 (2016), pp. 207–217.

[19] V. Melnykov, W.-C. Chen, and R. Maitra, *MixSim: An R Package for Simulating Data to Study Performance of Clustering Algorithms*, Journal of Statistical Software, 51 (2012), pp. 1–25.

[20] V. Melnykov and R. Maitra, *Finite mixture models and model-based clustering*, Statistics Surveys, 4 (2010), pp. 80–116.

[21] V. Melnykov and S. Michael, *Clustering Large Datasets by Merging K-Means Solutions*, Journal of Classification, 37 (2020), pp. 97–123.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[23] A. Rodriguez and A. Laio, *Clustering by fast search and find of density peaks*, Science (New York, N.Y.), 344 (2014), pp. 1492–1496.

[24] G. Schwarz, *Estimating the Dimension of a Model*, The Annals of Statistics, 6 (1978), pp. 461–464.

[25] L. Scrucca, *Identifying connected components in Gaussian finite mixture models for clustering*, Computational Statistics & Data Analysis, 93 (2016), pp. 5–17.

[26] ——, *A fast and efficient Modal EM algorithm for Gaussian mixtures*, Statistical Analysis and Data Mining: The ASA Data Science Journal, 14 (2021), pp. 305–314.

[27] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery, *Mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models*, The R Journal, 8 (2016), pp. 289–317.

[28] S. Vassilvitskii and D. Arthur, *k-means++: The advantages of careful seeding*, in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2006, pp. 1027–1035.

Supplementary Material

Exemplars are consistent mode estimators We here prove that the exemplars selected by the peak-finding criterion are consistent estimates of the true modes of the underlying GMM density. We also discuss the ability of the peak-finding criterion to control the types of exemplars through the parameters l and τ. This analysis is adapted from the theoretical work in [13].

We first prove the uniform consistency of the kernel density estimator for the underlying mixture density. The kernel function is defined as $K : \mathbb{R}^p \to \mathbb{R}_+$, such that $\int_{\mathbb{R}^p} K(u)du = 1$. The kernel used in this work is Gaussian. The Gaussian kernel is spherically symmetric, non-increasing and has exponential decay. As such, there exists a non-increasing function $k : \mathbb{R}_+ \to \mathbb{R}_+$ such that $K(u) = k(|u|)$ for $u \in \mathbb{R}^p$, and

$$k(t) \leq C_\rho \cdot \exp(t^\rho), \quad \text{for } t > t_0.$$

We bound the error in estimating the true mixture density $f(\boldsymbol{x})$ for every $\boldsymbol{x} \in \mathbb{R}^p$ as follows. As f is a mixture of Gaussian components, it is globally Lipschitz; that is, $\exists L > 0$ s.t. for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^p$, $|f(\boldsymbol{x}) - f(\boldsymbol{x}')| \leq L\|\boldsymbol{x} - \boldsymbol{x}\|$. A uniform kernel density estimation bound is provided by [14].

Lemma 1 (Theorem 2 of [14])**.** *There exists a positive constant C', depending on f and K, such that the following holds with probability at least $1-1/n$ uniformly in $h > (\log n/n)^{1/p}$.*

$$\sum_{\boldsymbol{x}\in\mathbb{R}^p} |f_h(\boldsymbol{x}) - f(\boldsymbol{x})| < C' \cdot \left(h + \sqrt{\frac{\log n}{n \cdot h^p}}\right).$$

We next define the modes of the underlying density.

Definition 1. *The modes of f is the finite set*

$$\mathcal{M} = \{\boldsymbol{x} : \exists r > 0, \forall \boldsymbol{x}' \in B(\boldsymbol{x}, r), f(\boldsymbol{x}') < f(\boldsymbol{x})\}.$$

As f is a mixture of Gaussian components, it is infinitely differentiable at all $\boldsymbol{x}$. Denoting the gradient and Hessian of f by ∇f and $\nabla^2 f$, then $\nabla^2 f$ is negative definite for all $\boldsymbol{x} \in \mathcal{M}$. This yields the following statement.

Lemma 2 (Lemma 5 of [9])**.** *There exists $r_M, \hat{C}, \check{C} > 0$ such that the following holds for all $\boldsymbol{x}^* \in \mathcal{M}$ simultaneously:*

$$\check{C} \cdot \|\boldsymbol{x}^* - \boldsymbol{x}'\|^2 \leq f(\boldsymbol{x}^*) - f(\boldsymbol{x}') \leq \hat{C} \cdot \|\boldsymbol{x}^* - \boldsymbol{x}'\|^2,$$

for all $\boldsymbol{x}' \in A_{\boldsymbol{x}^*}$. Here, $A_{\boldsymbol{x}^*}$ is a connected component of $\{\boldsymbol{x} : f(\boldsymbol{x}) \geq \inf_{\boldsymbol{x}'\in B(\boldsymbol{x}^*, r_M)} f(\boldsymbol{x}')\}$, which contains $\boldsymbol{x}^*$ and does not intersect with other modes.

This lemma states that the density in a neighborhood around the modes of f can be well-approximated by a quadratic function, and is useful for the later proofs.

Following [13], we now define a stronger notion of a mode that allows clearer analysis of the peak-finding criterion.

Definition 2. *A mode $\boldsymbol{x} \in \mathcal{M}$ is an $(r, \zeta, \nu)^+$-mode, if $f(\boldsymbol{x}) > f(\boldsymbol{x}') + \zeta$ for all $\boldsymbol{x}' \in B(\boldsymbol{x}, r)\backslash B(\boldsymbol{x}, r_M)$ and $f(\boldsymbol{x}) > \nu + \zeta$. Let $\mathcal{M}^+_{r,\zeta,\nu} \subseteq \mathcal{M}$ denote the set of $(r, \zeta, \nu)^+$-modes of f.*

If $\boldsymbol{x}^*$ is an $(r, \zeta, \nu)^+$-mode, the definition states that $f(\boldsymbol{x}^*) \geq \nu + \zeta$. Also, x^* is a maximizer of f in a larger ball of radius r by at least ζ for points outside the region of quadratic decay $B(\boldsymbol{x}_0, r_M)$, as defined in Lemma 2, i.e., $f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}') - \zeta$, $\forall \boldsymbol{x}' \in B(\boldsymbol{x}^*, r)\backslash B(\boldsymbol{x}^*, r_M)$.

The peak-finding criterion requires the user to specify a threshold l for the local density value and a threshold τ for the distance to nearest neighbor of higher local density. Take the set of instances selected by the peak-finding criterion as $\mathcal{E} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_\kappa\}$. We show that $\mathcal{E}$ contains unique and consistent estimates of the $(\tau + \epsilon, \zeta, l)^+$-modes of f.

Theorem 1 (Adapted from Theorem 2 of [13])**.** *Let $\boldsymbol{x}^* \in \mathcal{M}^+_{\tau+\epsilon,\zeta,l}$ be a $(\tau + \epsilon, \zeta, l)^+$-mode of f, where $\zeta, \epsilon > 0$. Let $h = h(n)$ be chosen such that $h \to 0$ and $\log n/(n \cdot h^p) \to 0$ as $n \to \infty$. Then, there exists a $C > 0$, depending on f and K, such that the following holds for n sufficiently large with probability at least $1 - 1/n$. There exists a unique $\boldsymbol{e} \in \mathcal{E}$ such that*

$$\|\boldsymbol{e} - \boldsymbol{x}^*\| < C\left((\log n)^{4/\rho} h^p + \sqrt{\frac{\log n}{n \cdot h^p}}\right).$$

Proof. In order to prove Theorem 1, we require the following lemma.

Lemma 3 (Lemma 4 of [13])**.** *Suppose the same setting as Theorem 1. Define*

$$\bar{r}^2 \coloneqq \max\left\{\frac{32\hat{C}}{\check{C}}(\log n)^{4/\rho} h^2, 17 \cdot C'\sqrt{\frac{\log n}{n \cdot h^p}}\right\}.$$

Suppose $\boldsymbol{x}^* \in \mathcal{M}$, and $\boldsymbol{x}^*$ is the unique maximizer of f on $B(\boldsymbol{x}^*, \bar{r})$. Define $\hat{\boldsymbol{x}} \coloneqq \arg\max_{\boldsymbol{x}\in B(\boldsymbol{x}^*,\bar{r})\cap \boldsymbol{X}} f_h(\boldsymbol{x})$, Then for n sufficiently large, with probability $1-1/n$ we have

$$\|\boldsymbol{x}^* - \hat{\boldsymbol{x}}\| < \bar{r}.$$

Suppose that $\boldsymbol{x}^* \in \mathcal{M}^+_{\tau+\epsilon,\zeta,l}$. Let $\boldsymbol{e} = \arg\max_{\boldsymbol{x}\in B(\boldsymbol{x}^*,\tau)\cap \boldsymbol{X}} f_h(\boldsymbol{x})$. We show that $\boldsymbol{e} \in \mathcal{E}$. By

Lemma 3, we have $\|\boldsymbol{x}^* - \boldsymbol{e}\| \leq \tilde{r}$ where

$$\tilde{r}^2 = \max\left\{\frac{32\hat{C}}{\check{C}}(\log n)^{4/\rho}h^2, 17 \cdot C'\sqrt{\frac{\log n}{n \cdot h^d}}\right\}.$$

It remains to show that $\boldsymbol{e} = \arg\max_{\boldsymbol{x}\in B(\boldsymbol{e},\tau)\cap \boldsymbol{X}} f_h(\boldsymbol{x})$. We have $B(\boldsymbol{e},\tau) \subseteq B(\boldsymbol{x}^*, \tau + \tilde{r})$. Choose n sufficiently large such that (i) $\tilde{r} < \epsilon$; (ii) $\sup_{\boldsymbol{x}\in\mathbb{R}^d} |f_h(\boldsymbol{x}) - f(\boldsymbol{x})| < \zeta/4$; and (iii) $\tilde{r}^2 < \zeta/(4\hat{C})$.

Now

$$\begin{aligned}
\sup_{\boldsymbol{x}\in B(\boldsymbol{x}^*,\tau+\tilde{r})\setminus B(\boldsymbol{x}^*,\tau)} f_k(\boldsymbol{x}) &\leq \sup_{\boldsymbol{x}\in B(\boldsymbol{x}^*,\tau+\tilde{r})\setminus B(\boldsymbol{x}^*,\tau)} f(\boldsymbol{x}) + \zeta/4 \\
&\leq f(\boldsymbol{x}^*) - 3\zeta/4 \\
&\leq f(\boldsymbol{e}) + \hat{C}\tilde{r}^2 - 3\zeta/4 \\
&< f(\boldsymbol{e}) - \zeta/2 \\
&< f_k(\boldsymbol{e}).
\end{aligned}$$

Therefore, we have $\boldsymbol{e} = \arg\max_{\boldsymbol{x}\in B(\boldsymbol{e},\tau)\cap \boldsymbol{X}} f_h(\boldsymbol{x})$. Furthermore, by (ii) we have that $f_h(x) > l - \zeta$. Hence, $\boldsymbol{e} \in \mathcal{E}$.

Now we show that it is unique. Suppose that $\boldsymbol{e}' \in \mathcal{E}$ such that $\|\boldsymbol{e}' - \boldsymbol{x}^*\| \leq \tau/2$. Then, both $\boldsymbol{e} = \arg\max_{\boldsymbol{x}\in B(\boldsymbol{e},\tau)\cap \boldsymbol{X}} f_h(\boldsymbol{x})$ and $\boldsymbol{e}' = \arg\max_{\boldsymbol{x}\in B(\boldsymbol{e}',\tau)\cap \boldsymbol{X}} f_h(\boldsymbol{x})$. However, choosing n sufficiently large such that $\tilde{r} < \tau/2$ yields $\boldsymbol{e} \in B(\boldsymbol{e}', \tau)$, implying $\boldsymbol{e} = \boldsymbol{e}'$.

□

Theorem 1 proves that the peak-finding criterion recovers the modes of a GMM density consistently.

Removal of the First Term from the Objective Function We provide a motivation for the removal of the first term from the objective discussed in Section 4.2.1. Consider again the original objective function before removing the first term

$$\min_{\{\boldsymbol{R}_{i\cdot}\in\Delta\}_{i=1}^n} \quad \sum_{i=1}^n\sum_{j=1}^\kappa r_{ij}\left[\log\left(\frac{r_{ij}}{\pi_j}\right) + b_{ij}\right],$$

where $b_{ij} = \frac{1}{2}d_{ij} + \frac{1}{2}\xi_j + \frac{1}{\kappa}\theta\delta_j$ is thus a constant determined by the data and the penalty. Further expansion yields

$$\min_{\{\boldsymbol{R}_{i\cdot}\in\Delta\}_{i=1}^n} \quad \left[\sum_{i=1}^n\sum_{j=1}^\kappa r_{ij}\log\left(\frac{r_{ij}}{\pi_j}\right) + \sum_{i=1}^n\sum_{j=1}^\kappa r_{ij}b_{ij}\right].$$

Now, we have that the first term

$$\sum_{i=1}^n\sum_{j=1}^\kappa r_{ij}\log\left(\frac{r_{ij}}{\pi_j}\right),$$

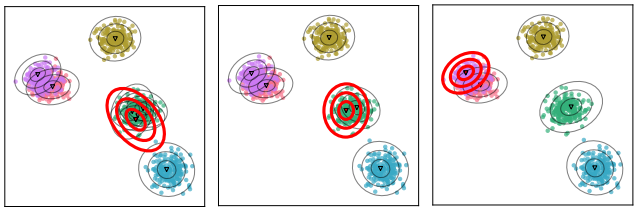


Figure 9: An illustrative example of the pruning block.

is a bounded function over the product of the n simplex spaces $\Delta \times \Delta \times \ldots \times \Delta$. Moreover,

$$\lim_{r_{ij}\to 0} r_{ij}\log\left(\frac{r_{ij}}{\pi_j}\right) = 0$$

$$\lim_{\pi_j\to 0}\sum_{i=1}^n r_{ij}\log\left(\frac{r_{ij}}{\pi_j}\right) = 0.$$

Therefore, the optimal solution, for the term $\sum_{i=1}^n\sum_{j=1}^\kappa r_{ij}\log\left(\frac{r_{ij}}{\pi_j}\right)$, must lie in a region where no $\pi_j = 0$ (in fact in a region where no $r_{ij} = 0$). As the arguments of the function are completely exchangeable and have equal importance, were say $\pi_1 = 0$, then it must be the case that $\pi_2 = 0$. The solution would thus reduce to $\pi_1 = \ldots = \pi_{\kappa-1} = 0$ and $\pi_\kappa = 1$.

Illustrative Example of the Pruning Block Figure 9 shows a worked example with five clusters. The colors represent the clusters; the contours represent the covariance matrices, with red contours indicating the component to be pruned in the next iteration.

Piecewise Linear Homotopy The parameter θ controls the amount of shrinkage on $\boldsymbol{\pi}$. We now provide details on how the trajectory of $\boldsymbol{\pi}$, as a function of θ, is computed. The objective function (4.2) is equivalent to

$$\min_{\{\boldsymbol{R}_{i\cdot}\in\Delta\}_{i=1}^n} \sum_{i=1}^n \boldsymbol{R}_{i\cdot}^T[\boldsymbol{D}_{i\cdot} + \boldsymbol{\xi} + \theta\boldsymbol{\delta}] = \sum_{i=1}^n \min_{\boldsymbol{R}_{i\cdot}\in\Delta} \boldsymbol{R}_{i\cdot}^T\boldsymbol{b}_i,$$

where $\boldsymbol{b}_i = \boldsymbol{D}_{i\cdot} + \boldsymbol{\xi} + \theta\boldsymbol{\delta}$, and we have incorporated the factor $\frac{1}{2}$ into the parameter θ. The solution to each individual problem is:

$$r_{ij}^* = \begin{cases} 1, & \text{if } j = \arg\min_{1\leq v\leq\kappa}\{b_v\}; \\ 0, & \text{otherwise.} \end{cases}$$

If the first exemplar $\boldsymbol{e}_1$ were to be pruned, we would require a θ value such that:

$$\begin{aligned}
b_{11} &\geq \min\{b_{12}, b_{13}, \ldots, b_{1\kappa}\} \\
b_{21} &\geq \min\{b_{22}, b_{23}, \ldots, b_{2\kappa}\} \\
&\vdots \\
b_{n1} &\geq \min\{b_{n2}, b_{n3}, \ldots, b_{n\kappa}\}
\end{aligned}$$

Therefore, the θ value should satisfy that

$$\begin{aligned} d_{11} + \xi_1 + \theta\delta_1 &\geq \min\{d_{12} + \xi_2 + \theta\delta_2, d_{13} + \xi_3 + \theta\delta_3, \ldots\} \\ d_{21} + \xi_1 + \theta\delta_1 &\geq \min\{d_{22} + \xi_2 + \theta\delta_2, d_{23} + \xi_3 + \theta\delta_3, \ldots\} \\ &\vdots \\ d_{n1} + \xi_1 + \theta\delta_1 &\geq \min\{d_{n2} + \xi_2 + \theta\delta_2, d_{n3} + \xi_3 + \theta\delta_3, \ldots\} \end{aligned}$$

To simplify notation, define $p_{ij}^1 = (d_{ij} + \xi_j) - (d_{i1} + \xi_1)$, for $j = 2, \ldots, \kappa$. We assume w.l.o.g. that $\delta_1 < \delta_2$ and $\delta_1 > \delta_j$ for $j \geq 3$. Then we can find the possible range of solutions for θ by considering the following set of intervals:

(6.3)
$$\theta \in \bigcap_{i=1}^{n} \left[\left(-\infty, \frac{p_{i2}}{\delta_1 - \delta_2} \right] \bigcup \left(\bigcup_{j=3}^{\kappa} \left[\frac{p_{ij}}{\delta_1 - \delta_j}, \infty \right) \right) \right]$$

If the interval solutions overlap, then the θ value that prunes the first exemplar is the lower bound of the overlapping interval. If the interval solutions do not overlap, resulting in $\theta \in \varnothing$, then the first exemplar will not be pruned.

By analogy, we can obtain the critical θ value for each exemplar, denoted by $\{\theta_1, \ldots, \theta_n\}$. Then with θ in (4.2) gradually increasing from 0, all the proportions in $\boldsymbol{\pi}$ change linearly. When θ reaches to $\min\{\theta_1, \ldots, \theta_n\}$, the exemplar with the minimal critical θ value will be pruned first, and its proportion value reduces to 0.

Trajectory of Component Proportions Figure 10 shows a toy example tracking the piecewise-linear trajectory of $\boldsymbol{\pi}$. Taking initially, two isotropic Gaussian clusters with four exemplars selected initially, the exemplars are labelled in decreasing order of $f_h(\boldsymbol{x}) \times \omega(\boldsymbol{x})$. As the iterations of the algorithm continue, the path of $\boldsymbol{\pi}$ is seen in the bottom row of figures to consecutively prune superfluous exemplars from the set. After two iterations, the optimal clustering is achieved.

Supplementary Experimental Results Figure 11 and Figure 12 present the analysis used to assess the impact of the bandwidth parameter and the impact of κ on the clustering quality, as done in Figure 7 and Figure 8 respectively, for the Ecoli, G2 and Wine datasets. While the Ecoli dataset is seen to be relatively sensitive to the parameter choice, the range of high-quality values allows for λ to be selected between 0 and 0.3, a wide range for users to exploit. The other two datasets exhibit perfect consistency for all values of λ, emphasising the robustness of the proposed approach. The Ecoli and G2 datasets are also robust to the choice of κ. While for the Wine dataset, the quality is seen

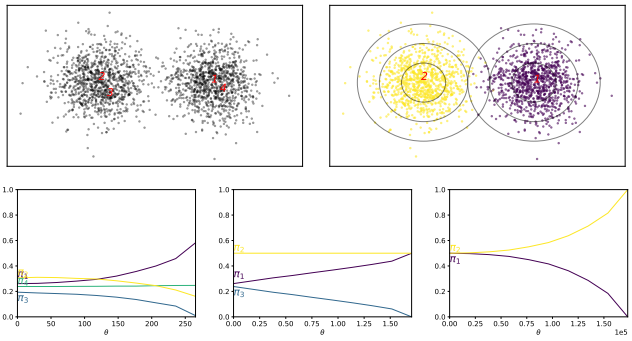


Figure 10: A toy example shows the piecewise-linear trajectory of $\boldsymbol{\pi}$. Top Left: The data and the four selected exemplars, labelled in decreasing order of $f_h(\boldsymbol{x}) \times \omega(\boldsymbol{x})$. Top Right: The final clustering obtained by REM. Bottom: The whole trajectory of $\boldsymbol{\pi}$, as a function of θ, in each REM iteration. After the first iteration, exemplar $\boldsymbol{e}_3$ is pruned; after the second iteration, exemplar $\boldsymbol{e}_4$ is pruned. The bottom right panel shows that θ needs to be very large to merge two true cluster centers.

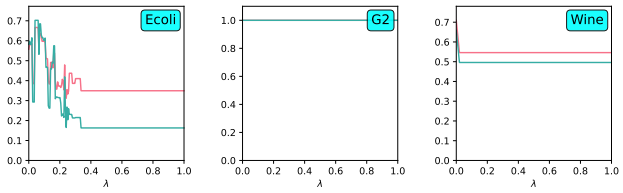


Figure 11: The performance of the REM algorithm evaluated by the ARI (pink) and NMI (blue) as the parameter h changes.

to degrade as κ increases, it should be noted that the decision graph included in Figure 6 points users to values that return high-quailty clusterings.

The results for the ablation study for the G2, Iris, Satellite, Seeds and Wine datasets are shown in Table 5. The strength of the REM method is again demonstrated. The peak-finding initializations are seen to significantly outperform the random initialization used in M1. Furthermore, the peak-finding initializations are seen to complement the pruning method introduced in this work.

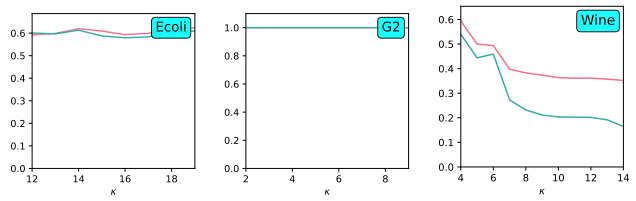


Figure 12: The performance of the REM algorithm evaluated by the ARI (pink) and NMI (blue) as the number of selected exemplars changes.

	Metric	G2	Iris	Sat.	Seeds	Wine
M1	ARI	0.431	0.795	0.516	0.475	0.383
	NMI	0.522	0.808	0.572	0.555	0.471
M2	ARI	**1.000**	0.899	0.463	0.640	0.463
	NMI	**1.000**	0.879	0.559	0.670	**0.501**
M3	ARI	0.000	0.664	0.492	0.641	0.456
	NMI	0.000	0.730	0.597	0.671	**0.501**
M4	ARI	**1.000**	0.870	**0.524**	0.715	0.000
	NMI	**1.000**	0.883	**0.578**	0.737	0.000
REM	ARI	**1.000**	**0.904**	**0.524**	**0.766**	**0.534**
	NMI	**1.000**	**0.900**	**0.578**	**0.744**	**0.501**

Table 5: Clustering results on the real datasets by ablation methods. The best results are highlighted in bold.