International Journal of Data Science and Analytics (2026) 21:38
https://doi.org/10.1007/s41060-025-00912-x

REGULAR PAPER

Nested Gaussian mixture model: clustering data from generalized mixture distributions

Fei Wang[1] · **Le Li**[1] · **Pasi Fränti**[1]

Received: 19 June 2025 / Accepted: 9 October 2025

Abstract

Hierarchical clustering and the Gaussian mixture model are two important means of cluster analysis. As a generative model, a well-trained Gaussian mixture model can naturally characterize the distribution of data consisting of Gaussian clusters and thus gain the ability to handle incremental data. However, the validity of Gaussian mixture model relies on the consistency between data distribution and model assumption. That is, it requires particular knowledge about the prior distribution of a dataset for the user to decide whether Gaussian mixture model is competent since GMM fails to model data from non-Gaussian mixture distribution. By contrast, agglomerative nesting is a nonparametric, hierarchical clustering algorithm that does not require prior knowledge. In practice, agglomerative nesting tends to fluctuate drastically in performance with different similarity metrics and requires a lot of additional computing resources as the sample size grows. This paper proposes an arbitrary mixture model for characterizing data from unknown mixture distribution. In the absence of information about the prior distributions of clusters, we leverage a nested mixture model to approximate the unknown mixture distribution. This is achieved by decomposing the unknown distributed clusters and remodeling them with an equal number of Gaussian mixtures. Particularly, we implement the two successive procedures of the method achieved by using the EM algorithm with incomplete covariance and SingleLink agglomeration on a compressed similarity matrix. Experimental results demonstrate that our method shows clear advantages over the Gaussian mixture model and agglomerative nesting in both runtime and performance, improving clustering accuracy by 12–25% across standard evaluation indices and reducing the cluster number error to 12, compared with 70 and 151 for competing Gaussian mixture-based methods.

Keywords Hierarchical clustering · Generative model · Agglomerative nesting · Gaussian mixture model · EM algorithm

1 Introduction

Clustering methods based on different ideas have achieved considerable progress in the past decades [4], among which hierarchical clustering (HC) approaches [23, 26, 27] and model-based methods [5] have drawn wide attention. In addition to general data analysis, clustering has also been adopted in application domains such as the Internet of Things and flying ad hoc networks, where cluster-based routing is exploited for energy-efficient and delay-aware communication [30–32].

Compared with flat labelings given by other clustering techniques, such as partition-based [1] or density-based methods [29], HC usually constructs a hierarchy of flat labelings. In particular, hierarchical agglomerative clustering (HAC) iteratively merges clusters with the highest similarity to build a clustering hierarchy, where each level represents a flat clustering. The hierarchy construction often suffers from high time and space complexity, $O(n^3)$ and $O(n^2)$, respectively, where n represents the size of the used dataset. The performance of AGNES [34] can differ significantly when applying distinct linkage functions, such as SingleLink [33], CompleteLink, AverageLink, and Ward [28, 39, 40]. SingleLink usually performs poorly in practice due to its oversensitivity to noise; however, it excels in the potential for detecting clusters of complex shapes. This resulted in studies of improved SingleLink [9, 10, 18, 36]. In comparison,

✉ Fei Wang
fei.wang@uef.fi

Le Li
leli@cs.uef.fi

Pasi Fränti
franti@cs.uef.fi

[1] School of Computing, University of Eastern Finland, 80100 Joensuu, Finland

Published online: 02 December 2025

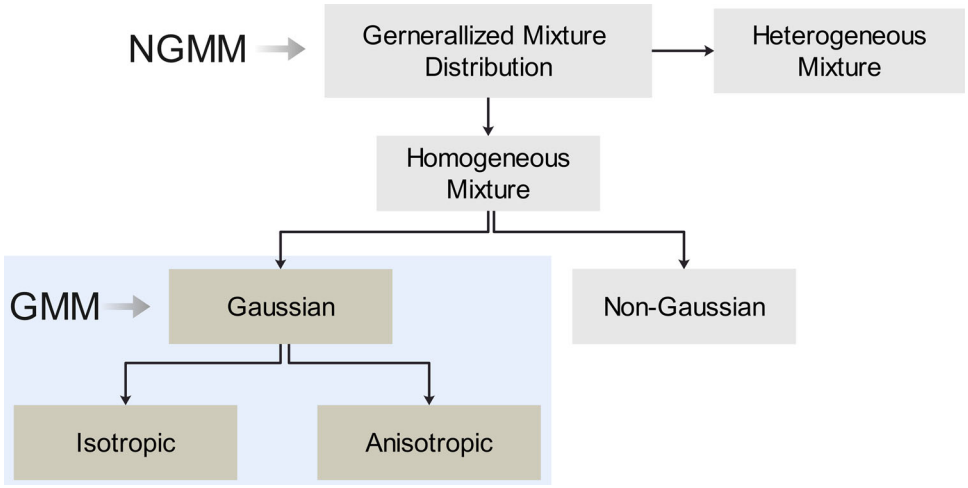


Fig. 1 The structure of generalized mixture distribution. GMM is limited to Gaussian mixture distributions, while NGMM is able to model data from generalized mixture distributions

although other linkages are insensitive to noise, the preferences of generating spherical clusters and breaking down clusters of relatively large size limit the general performance.

From the perspective of statistical machine learning, exploring and estimating the latent distribution model from finite samples is one of the most critical objectives. In general, the distribution characteristics of arbitrary datasets can be covered with the generalized mixture distribution. Figure 1 shows the structure of the generalized mixture distribution.

Gaussian mixture model (GMM), as a representative model-based method, aims to model the clusters using Gaussian components [6, 7, 13, 25]. Considering the generalized mixture distribution of real-world data, the GMM application scenarios are limited to the Gaussian mixture distribution. It is also necessary to acquire specific knowledge about the prior distribution of the dataset to judge whether GMM can be a desirable choice. Besides, the iterative computation of the full covariance matrix for each Gaussian component is computationally costly, especially on high-dimensional data. Using the incomplete form of the covariance matrix can significantly reduce the time cost; however, it is at the expense of the general adaptability of GMM, since this will impose a stricter assumption that all clusters are isotropically Gaussian distributed.

Particularly, two artificial datasets were generated to illustrate the limitations of GMM. As shown in Fig. 2a and b, M3, consisting of a Gaussian cluster, a circular cluster, and a moon-like cluster, corresponds to a heterogeneous mixture distribution, while M2, consisting of two moon-like clusters, is an instance of the homogeneous non-Gaussian mixture distribution. It can be observed that GMM fails in both cases according to Fig. 2c and d.

So this paper aims to explore a more flexible and adaptable model-based method for effectively modeling generalized mixture distributions and correctly revealing the latent cluster structures.

The main contributions of the present work can be briefly summarized as follows.

- This work first formalizes the idea of the generalized mixture model for arbitrary mixture distributions.
- In the absence of information on the prior distribution of the mixture components, this paper proposes the nested Gaussian mixture model (NGMM), which simulates the unknown distribution of a cluster with an independent GMM.
- Due to the NGMM's two-layer hierarchical design, this paper presents a concise 3-stage approach that encompasses distribution decomposition, similarity modeling and analysis, and hierarchical structure reconstruction to semiparametrically estimate an NGMM.

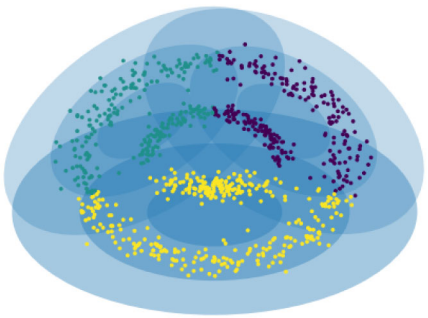
2 On the integration of hierarchical ideas and generative models

There are usually two directions for constructing a hierarchy of generative models: bottom up and top down.

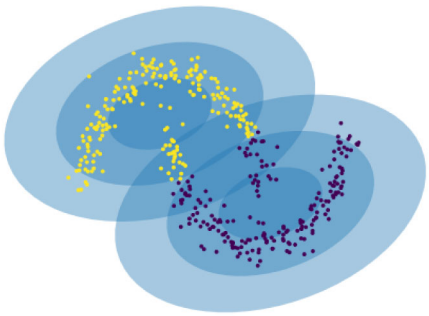
The approach of Goldberger et al [19] involves bottom-up hierarchy, which aims to reduce a given large mixture of Gaussians into a smaller mixture while still preserving the component structure of the original model. Let f be the probability density of a Gaussian mixture with k components, the method collapses the Gaussian component of f into $k'(k' < k)$ groups, represented by g, and represents each group with a new single Gaussian density by minimizes the distance measure between the two Gaussian mixtures: $d(f, g) = \sum_{i=1}^{k} \alpha_i \min_{j=1}^{k'} KL(f_i || g_j)$, where α_i is the prior probability of ith components of f and $KL(\cdot)$ is the Kullback–Leibler divergence. In an ideal situation, the new mixture will fully reveal information about the hierarchical structure of the original mixture components. However, this

(a) Dataset M3 from a heterogeneous mixture distribution (3 different shaped clusters).

(b) Dataset M2 from a non-Gaussian distribution (2 moon-like clusters).



(c) The estimated 3-component Gaussian mixture and the resulting 3 clusters (with 3 different colors).



(d) The estimated 2-component Gaussian mixture and the resulting 2 clusters (with 2 different colors) .

Fig. 2 The limitation of GMM for handling mixture distributions involving non-Gaussian components: The GMMs were estimated with EM using components consistent with ground truth. However, the detected clusters (with different colors) clearly disagree with the true distributions

is not the case when the real target clusters we want to capture by g are non-Gaussian, e.g., the clusters are entangled with each other (as shown in the examples in Fig. 2). This approach is advantageous in reducing model complexity while preserving Gaussian component information. However, its reliance on Gaussian assumptions limits its applicability to entangled or non-Gaussian clusters.

In [3], Bahrololum et al. exploited another top-down method, the Hierarchical Gaussian Mixture Model (HGMM), for learning patterns of normal and intrusive activities. In their work, the HGMM is considered a general GMM, and all Gaussian components in the local GMM belong to a root node that is also represented by a Gaussian component. During training, the observation vectors of each local Gaussian component are pooled to estimate the new parameter of the root nodes. This approach can be understood as another estimation of the GMM with the Gaussian mixture components as the observations. Similarly to the above methods, the construction of the hierarchical structure of HGMM relies on the assumption of Gaussian distribution of the data. The method is effective for hierarchical modeling of structured Gaussian data and has practical utility in anomaly detection. Yet, it fails to generalize to non-Gaussian data since every level of the hierarchy is restricted to Gaussian mixtures.

In work [2], the authors presented a top-down HGMM-based algorithm that can automatically choose the initialization, number of clusters, and covariance constraints. Specifically, the implementation of AutoGMM initializes the parameters of initial GMMs using agglomerative clustering with different linkages and all 4 types of covariance. This method reduces the risk of an inappropriate model caused by the pre-assumption of covariance type and provides more reasonable options for parameter initialization. However, it also adds to the considerable computational cost of model selection. The hierarchical version of this algorithm implements AutoGMM recursively (in a top-down manner) on each cluster determined by the AutoGMM process to identify potential subclusters. This extension is useful when studying data consisting of Gaussian clusters with a natural hierarchy. The restriction is that it is unable to reveal the cluster structure of non-Gaussian clusters as well as the natural hierarchies within non-Gaussian clusters. AutoGMM improves model selection robustness and reduces sensitivity to initialization, making it practical for Gaussian datasets with complex covariance structures. Nevertheless, the recur-

sive estimation greatly increases computational cost, and the approach remains ineffective for non-Gaussian clusters.

The idea of hierarchical generative models can also be seen in the research field of point cloud registration. In work [15], point cloud data are modeled via a GMM-Tree, which is built in a top-down recursive fashion from small-sized Gaussian mixtures. On this basis, the authors leveraged the Hierarchical Gaussian Mixture Representation (HGMR) using a novel optimization criterion that adaptively finds the best scale to perform data association between spatial subsets of point cloud data [16]. Each level of the hierarchy constructed by this type of method consists of a Gaussian mixture constructed from the base of the adjacent Gaussian mixture of the upper level. These methods demonstrate the flexibility of hierarchical GMMs in spatial data, and their multi-scale representation improves registration accuracy. However, they remain bound to Gaussian-shaped assumptions at each level and thus cannot capture more general cluster structures.

In [38], a stratification-based semi-supervised clustering algorithm was proposed for handling datasets with arbitrary shapes. The method first applies a k-means-like decomposition initialized by randomly scattered stratified seeds and then constructs a hierarchy by exploiting the relationships among these seeds. This two-stage design enables the algorithm to better capture clusters of complex shapes and improves robustness to distributional irregularities. However, as a semi-supervised method, it relies on partial label information to guide stratification and seed selection, which restricts its applicability in fully unsupervised scenarios.

The above unsupervised methods share the property that each level in the hierarchy remains to be a Gaussian mixture model, whether the hierarchy is constructed from top to bottom or bottom to top. This property dictates that even with the introduction of hierarchies, these models are still incompetent to be used in clustering non-Gaussian data. In contrast, the hierarchy constructed in this work contains a mixture of Gaussian mixtures rather than a single GMM.

3 Problem formalization

3.1 Generalized mixture model

Let $X = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^d$ represents a set of n data objects sampled from the mixture of m probability distributions. It can be concluded that GMM fails to model the latent distribution of X as long as some of the distributions do not agree with the Gaussian distribution. Assuming the prior distribution of X is known in advance, instead of pre-assuming the sharing Gaussian distribution for all clusters, we can model the dataset as a generalized mixture distributions. That is, we will treat each cluster as sampling from a particular distribution that is in accordance with the prior knowledge, which may be either Gaussian or non-Gaussian. Therefore, the generalized mixture model $\mathcal{M}_\Theta$ can be formalized as the mixture of m particular probability models:

$$p_{\mathcal{M}_\Theta}(x|\Theta) = \sum_{j=1}^{m} \beta_j \, p(x|\theta_j) \tag{1}$$

where β_j is the mixture coefficient (prior probability), and $p(x|\theta_j)(j \in [1, m])$ is the probability density function of the j_{th} components of $\mathcal{M}_\Theta$.

Note that the generalized mixture model requires information about the prior distribution of the clusters. However, often we know little or nothing about such information. It is better, if possible, to solve the problem without using any prior knowledge. This can be a tricky problem if we consider solving it with conventional ideas such as EM algorithm.

3.2 Nested Gaussian mixture model

In practice, it is possible to decompose a cluster into a group of fine-grained, tiny clusters (or sub-clusters) of homogeneous distribution. For instance, the original distribution shown in Fig. 2 can be divided into 3(2) groups of spherical sub-clusters. The observation can also be extended equally to Euclidean space of higher dimensionality. This gives rise to the idea of approximating the unknown distribution of each independent cluster with a homogeneous mixture model that can be well explained. We first discuss the feasibility of the solution from the perspective of partition. Let $\mathbb{D}$ represent the true partition containing m clusters:

$$\mathbb{D} = \{D^1, D^2, ..., D^m\} \tag{2}$$

where D^j is the jth cluster.

By separating the partition $\mathbb{D}$ into m sub-partitions among which all the sub-clusters share similar distribution characteristics, we have a two-level partition, denoted as $\mathbb{D}_S$, as follows:

$$\begin{aligned} &\mathbb{D}_S = \{D_S^1, D_S^2, ..., D_S^m\}, where \\ &D_S^1 = \{C_1^1, C_2^1, ..., C_{c_1}^1\}, \\ &\quad\vdots \\ &D_S^m = \{C_1^m, C_2^m, ..., C_{c_m}^m\} \end{aligned} \tag{3}$$

where $D_S^j (j \in [1, m])$ is the j_{th} sub-partition consisted of c_j sub-clusters such that $D_S^j = \bigcup_{i=1}^{cj} \bigcup_{x \in C_i^j} \{x\} = \bigcup_{x \in D^j} \{x\} = D^j$. Particularly, C_i^j $(i \in [1, c_j])$ denotes the i_{th} sub-cluster of D_S^j.

Obviously, $\mathbb{D}_S$ comprises some extra information compared with $\mathbb{D}$. Note that the information about cluster structure implied in $\mathbb{D}_S$ is consistent with that given in $\mathbb{D}$.

It means that we can solve $\mathbb{D}_S$ instead of directly solving the partition $\mathbb{D}$.

To make the problem solvable, we can relax the problem by assuming Gaussian distribution for all the sub-clusters and dropping the information about the cluster structure within $\mathbb{D}_S$, leading to a structure-free partition $\mathbb{D}_u$.

Definition 3.1 *(Gaussian decomposition)* A Gaussian decomposition is a structure-free partition $\mathbb{D}_u$ such that: $\mathbb{D}_u = \bigcup_{j=1}^{m} \bigcup_{i=1}^{c_j} \{C_i^j\}$.

Evidently, $\mathbb{D}_u$ is not unique depending on the specific form $\mathbb{D}_S$, and according to $\mathbb{D}_u$, we can determine a GMM that is viewed as structure-free. We call $\mathbb{D}_u$ a 'Gaussian decomposition' since it implies the idea of 'breaking up the whole into parts.' In this way, $\mathbb{D}_S$ can be reconstructed by recovering the structural information on $\mathbb{D}_u$ through gathering the related 'parts.' Once $\mathbb{D}_S$ is reconstructed, it determines a nested Gaussian mixture model that stands for an approximation of the unknown mixture distribution of X.

Inspired by this, we consider modeling an unknown mixture distribution using a two-level nested Gaussian mixture model (NGMM) where each component of the nested mixture model is a separate Gaussian mixture. By approximating the unknown component distributions Θ in Eq. 1 with m Gaussian mixture distributions, the resulting structure can be formalized as a two-level NGMM:

$$p(x|\mathbb{M}) = \sum_{j=1}^{m} \beta_j \, p(x|\mathcal{M}_j) \tag{4}$$

where β_j is the first-level mixture coefficient and $p(x|\mathcal{M}_j)$ denotes the j_{th} Gaussian mixture consisted of c_i Gaussian components:

$$p(x|\mathcal{M}_j) = \sum_{i=1}^{c_i} \alpha_i \, p(x|\mu_i, \Sigma_i) \tag{5}$$

where α_i is the second-level mixture coefficient, $p(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}$ is the probability density function of the i_{th} Gaussian component of the mixture.

The EM algorithm is widely used for probability models with latent variables. In contrast, NGMM not only incorporates a latent variable but also an additional conditioned latent variable, necessitating extra hyperparameter specification. Thus, we address NGMM distinctively through a special strategy: decomposition and reconstruction.

4 Determination of an NGMM

The overall algorithm for constructing an NGMM consists of 3 main steps: 1. solve a Gaussian capture and the corresponding Gaussian decomposition; 2. properly model the similarity between Gaussians within the Gaussian capture, based on which analyze the latent structure among Gaussians; 3. extract nested structure from the hierarchy to form the finial NGMM as a mixture of GMMs. Algorithm 1 provides a summary of the procedures to solve an NGMM.

Algorithm 1 3-stage solution for an NGMM

Input: X, covariance type = 'spherical', 'diagonal' for high-dimensional data
Output: NGMM

1. Decomposition: Solve a Gaussian capture by minimizing $J(t)$ (Eq. A1);
2. Similarity modeling and analysis:
2.1: Modeling similarity over Gaussians according to CSR similarity (Eq. 8);
2.2: Applying Single-linkage Agglomeration (Algorithm 2) to obtain two types of data structure H, T.
3. Hierarchical structure reconstruction through one of the following strategies: (1) Reconstruct NGMM by observing the dendrogram from H (refer to Fig.3 (c)),T (refer to Fig.4); (2) Or simply split H according to prior knowledge, for example, the last but 1 layer of H (H[-2]) corresponds to an NGMM with 2-GMM (or 2-cluster) solution.

4.1 Capture a Gaussian decomposition

Obviously, an NGMM determines a two-level partition in the form of $\mathbb{D}_S$. The cluster structure we are seeking is contained in the first-level mixture of NGMM. Conversely, once $\mathbb{D}_S$ is determined, the NGMM is also determined. So we can first determine a Gaussian decomposition $\mathbb{D}_u$ and then manage to recover the information about cluster structure in the form of $\mathbb{D}_S$. Thus, the first task is to find a structure-free Gaussian mixture model that satisfies our request. Corresponding to the Gaussian decomposition, we define the Gaussian capture to describe a group of Gaussian components that can separate original data into fine-grained Gaussian distributions.

Let $\mathcal{M}_t$ be a Gaussian mixture model estimated from X consisting of t Gaussian components. Let $z_l \in \{1, 2, ..., t\}$ be a random variable indicating the component of the model $\mathcal{M}_t$ from which a sample $x_l \in X$ is generated, the partition $\mathbb{C}_t = \{C_1, C_2, ..., C_t\}$ is derived by considering the maximum posterior probability for each $x_l \in X$: $max_i\{p(z_l = i|x_l)|(i = 1, 2, ..., t)\}$.

Definition 4.1 *(Gaussian capture)* A Gaussian capture is a Gaussian mixture model such that: for any $C_l \in \mathbb{C}_t, C_l \subset D^j$ $(j \in [1, m])$.

A Gaussian capture demands that all the objects in $C_l (l \in [1, t])$ should entirely belong to one of the true clusters. In this way, according to Definition 3.1, $\mathbb{C}_t$ is exactly a Gaussian decomposition, which means that a Gaussian capture uniquely corresponds to a Gaussian decomposition.

On the other hand, a Gaussian capture contains no information about the cluster structure and can be viewed as a structure-free GMM since it always decomposes the cluster distributions into fine-grained Gaussian components much more than the true number of clusters. It also means that Gaussian capture is not unique depending on the granularity of the Gaussian decomposition. By contrast, a general GMM is structured as it directly produces cluster assignation.

Based on the above analysis, we can find a Gaussian capture as a prerequisite for determining an NGMM. It is achieved with the help of some well-established techniques [5, 7, 22] for estimating structured GMMs despite there being some differences between the objectives.

Particularly, we also use the EM algorithm, a well-founded statistical algorithm, to seek a Gaussian capture through an iterative process. The general purpose of the EM algorithm is to solve GMMs for data from Gaussian mixture distributions. EM requires the number of mixture components to be specified in advance, which should essentially match the actual number of m in the cluster.

By contrast, our purpose of finding a Gaussian capture differs from that of estimating a GMM. In order to capture a Gaussian decomposition, we always need more Gaussian components than the true number of clusters m.

Assume that $\mathcal{M}_t$ is the best-fitting GMM containing t Gaussian components, i.e., with maximum likelihood. Theoretically, as t grows up to m, $\mathcal{M}_t$ will gradually satisfy the requirement of a Gaussian capture. The extreme case is that when $t = n$, $\mathcal{M}_t$ is a Gaussian capture. Furthermore, if $\mathcal{M}_z$ is a Gaussian capture, $\mathcal{M}_t$ is also a Gaussian capture when $t > z$. Thus in practice, we only need to specify a large enough t to get a Gaussian capture.

The time complexity of finding Gaussian capture is $O(c * t * n * d^2)$, $O(c * t * n * d)$, and $O(c * t * n)$ when using complete, diagonal, and spherical covariance, respectively, where c is a constant indicating the total iterations of EM, n, and d represent the size and dimension of the data X, respectively. It indicates that when dealing with high-dimensional data, the computational costs will rise significantly. As for GMM, assuming an incomplete covariance can reduce the dependency on d. However if users make assumptions about the GMM's covariance matrices, they risk inappropriate model restriction. By contrast, NGMM does not suffer the limitation due to its nested structure. Specifically, assuming an incomplete covariance may lead to an increase in t which means that more Gaussian components are needed for well simulating the unknown distribution of a cluster.

The number of Gaussian entities t needs to be specified to obtain a Gaussian capture, but our framework is relatively insensitive to this choice. To guide the selection, we introduce in Appendix A (Fig. 8) a reference criterion $J(t)$, which balances model likelihood with complexity and provides principled reference values for t (see Fig. 9 for examples with different covariance assumptions). More importantly, even when t is oversized, the subsequent hierarchical reconstruction and steady-tree extraction automatically recover the appropriate number of clusters, making NGMM robust to the initial choice of t.

4.2 Structure information recovery

For the purpose of reliable reconstructing structural information over a valid Gaussian capture, we need to ensure the reconstructed structure can reveal arbitrary non-Gaussian mixture distributions.

Motivated by the idea of SingleLink method in chaining similar points into complex shaped clusters, we manage to extend it from concrete data points to more abstract Gaussian objects. This way, an effective method is required to reveal as well as strengthening the similarity among the closest Gaussian pairs within a Gaussian capture.

Therefore, we propose the Cumulative Cross Responsibility (CCR) to model the similarity over Gaussians, which considers the mutual interaction between a Gaussian capture and the corresponding Gaussian decomposition.

Let $\hat{\mathcal{M}} = \bigcup_j^t \{G_j\}$ be a Gaussian capture with t Gaussian components, $\hat{\mathbb{C}}_t = \bigcup_j^t \{C_j\}$ is the Gaussian decomposition derived from $\hat{\mathcal{M}}$, where C_j is determined by G_j as every point in C_j has the largest responsibility to G_j ($\forall x_l \in C_j, j = argmax_k p(z_l = k|x_l)$).

We formulate CCR as the total responsibility from a cluster C_i to a Gaussian G_j.

$$p(G_j|C_i) = \sum_{x_l \in C_i, j \neq i} p(z_l = j|x_l) \tag{6}$$

where $p(z_l = j|x_l)$ is the responsibility of x_l to G_j:

$$\begin{aligned} p(z_l = j|x_l) &= \frac{p(z_l = j)p(x_l|z_l = j)}{\sum_{k=1}^t p(z_l = k)p(x_l|z_l = k)} \\ &= \frac{\frac{|C_j|}{n} \frac{1}{\sqrt{2\pi|\Sigma_j|}} e^{(x_l-\mu_j)^T \Sigma_j^{-1} (x_l-\mu_j)}}{\sum_{k=1}^t \frac{|C_k|}{n} \frac{1}{\sqrt{2\pi|\Sigma_k|}} e^{(x_l-\mu_k)^T \Sigma_k^{-1} (x_l-\mu_k)}} \end{aligned} \tag{7}$$

Since $i \neq j$, the approach indicates cross-similarity from C_i to all other Gaussians excluding G_i. The cumulative nature of the CCR similarity significantly reduces the sensitivity of

the original single-link method to noise or outliers, which frequently cause erroneous cluster merge.

Note that, $p(G_j|C_i)$ gives an unidirectional reliance from C_i to G_j. In order to ensure the symmetry of the similarity metric, we can consider the mutual reliance to measure the general closeness between G_i and G_j:

$$S_{i,j} = p(G_j|C_i) + p(G_i|C_j) \tag{8}$$

For common AGNES clustering approaches, a full hierarchy consisting of n levels (flat cluster assignments) will finally be constructed after the agglomerative [40] (or divisive [8, 23]) process. Flat cluster assignation can be obtained by horizontally cutting the hierarchy [21] or by applying some automatic techniques [9, 18, 36]. The above methods all need to build an nn similarity matrix for the original data at the beginning, resulting in a high cost of storage space and running time.

Different from the above methods, we construct a hierarchy for the abstract Gaussian objects to intuitively reveal the nested structure. It means that only a t by t similarity matrix is need to be built (t is usually much smaller compared with variable n). As a result, the space and time complexity of constructing the similarity matrix as well as the subsequent SingleLink procedure can be significantly reduced.

On the basis of S, the corresponding hierarchy, denoted as H, can be established by applying a SingleLink agglomerative procedure, among which each layer contains a flat cluster assignment and the related nested structure .

The pseudocode of the SingleLink Agglomeration over Gaussians is given in Algorithm 2:

The input S is a $t \times t$ similarity matrix constructed according to Eq. 8, and the (i, j) element of S represents the similarity between a pair of abstract objects.

The output H represents a hierarchy that is similar to the common agglomerative clustering algorithms. H[0] is the initial level of the hierarchy containing t clusters. In each while loop, the clusters with the highest similarity will be merged into a cluster. Thus H[1] contains $t - 1$ clusters. After $t - 1$ round, H[t-1] contains only one clusters. Lines 10–11 of Algorithm 2 implement the merge process. Figure 3b shows the dendrogram that visualizes the hierarchy constructed according to the output H on dataset M2.

For automatically extracting steady trees as clusters, another data structure T is required to record 5 types of the necessary information. In the kth while loop, the kth row of T, T[k-1], stores the following information about the current tree: left child, right child, parent, birth time, and life span. It is implemented in the *if-else conditions* in lines 12-24. The last 5 columns of the table in Fig. 4 show an instance of output T on dataset M2, while the tree plot shows an illustration of extracting 'steady trees'.

Algorithm 2 SingleLink Agglomeration

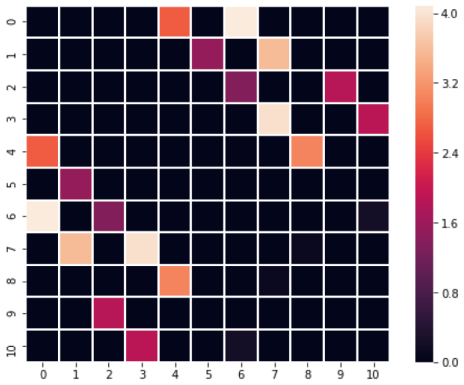
Input: S
Output: H, T
1: Initialize $t \times t$ matrix H $= -1$
2: H[0] $= 0, 1, 2, ..., t-1$
3: Initialize $(t-1) \times 5$ matrix T $= -1$
4: Initialize $N = 0$, $T_{No} = t$
5: **while** $N < t - 1$ **do**
6: s=$max(SM_u)$, i, j=$argmax(SM_u)$.
7: No_i = H$[N, i]$, No_j = H$[N, j]$
8: $IndexI = where$(H$[N]$ == No_i)
9: $IndexJ = where$(H$[N]$ == No_j)
10: H$[N+1]$ = H$[N]$
11: H$[N+1, IndexI] = T_{No}$, H$[N+1, IndexJ] = T_{No}$
12: **if** $No_i < t$ and $No_j < t$ **then**
13: T$[N, :] = No_i, No_j, -1, s, -1$
14: **else if** $No_i < t$ **then**
15: T$[N] = No_i, No_j, -1,$ T$[No_j - t, 3], -1$
16: T$[No_j - t, 2] = T_{No}$
17: **else if** $No_j < t$ **then**
18: T$[N] = No_j, No_i, -1,$ T$[No_i - t, 3], -1$
19: T$[No_i - t, 2] = T_{No}$
20: **else if** $No_i \neq No_j$ **then**
21: T$[No_i - t, 2] = T_{No}$, T$[No_j - t, 2] = T_{No}$
22: T$[No_i - t, 4]$ = T$[No_i - t, 3] - s$
23: T$[No_j - t, 4]$ = T$[No_j - t, 3] - s$
24: T$[N] = No_i, No_j, -1, s, -1$
25: **end if**
26: $N = N + 1$, $T_{No} = T_{No} + 1$
27: $SM[i, j] = 0$
28: **end while**

The pseudocode of the agglomerative procedure is given in Algorithm 2 of appendix. The agglomerative procedure uses S as the input and constructs a hierarchy H as output. In the while loop (lines 5-28), a pair of objects with the highest similarity is merged each round. The process consists of $t-1$ rounds in total, after which all objects will be merged. As one of the outputs, H contains t possible assignments of the abstract Gaussian objects, which corresponds to t NGMM solutions with 1 to t GMM components.

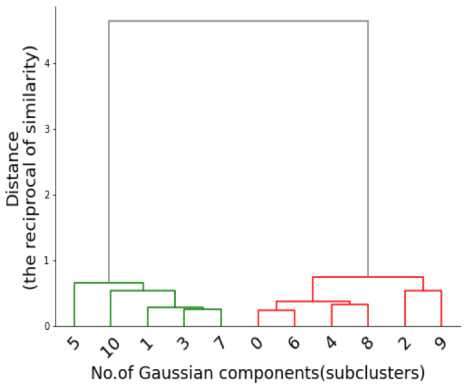
Each assignment contains information about the two-level cluster structure of the Gaussian decomposition and the nested structure of the Gaussian capture. For example, the last but two rows of H contain the 2-cluster solution with respect to the abstract objects. This means that all subclusters in the Gaussian decomposition $\hat{\mathbb{C}}_t$ are divided into two groups in the form of $\mathbb{D}_S$. Correspondingly, by correlating related components of the Gaussian capture $\hat{\mathcal{M}}$ into two Gaussian mixture models, the NGMM is also determined. Note that the dendrogram implied in the hierarchy H can be explicitly established to provide a decision basis for splitting the hierarchy (see Fig. 3b).

Figure 3 presents a complete example to illustrate the determination process and decision basis for an NGMM. The Gaussian capture with 11 ellipsoid components in Fig. 3a is used to compute a similarity matrix S. Figure 3b shows the similarity heatmap of S. From the heatmap, we can observe a

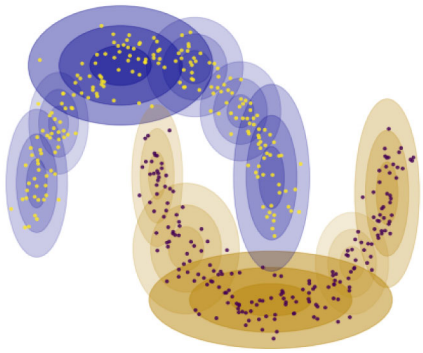
(a) Ellipsoid Gaussian capture with 11 components on M2



(b) The similarity matrix S over Gaussian pairs.



(c) The dendrogram built on S, where a 2-cluster structure can be easibly identified.



(d) The final NGMM consisted of two GMM components.

Fig. 3 Determination of NGMM by splitting the dendrogram. The compressed similarity matrix S is computed according to the Gaussian capture and Gaussian decomposition given in Fig. 9b

clear relation (close or distance) between each pair of components of the Gaussian capture(or Gaussian decomposition). Figure 3c shows the dendrogram constructed from S through a SingleLink procedure, among the levels the two-cluster solution intuitively stands out of all the possible divisions. As shown in Fig. 3d, according to the information about cluster structure derived from the dendrogram, an equivalent NGMM containing two Gaussian mixtures is established over the Gaussian capture. The target clusters are recovered from the Gaussian decomposition at the same time.

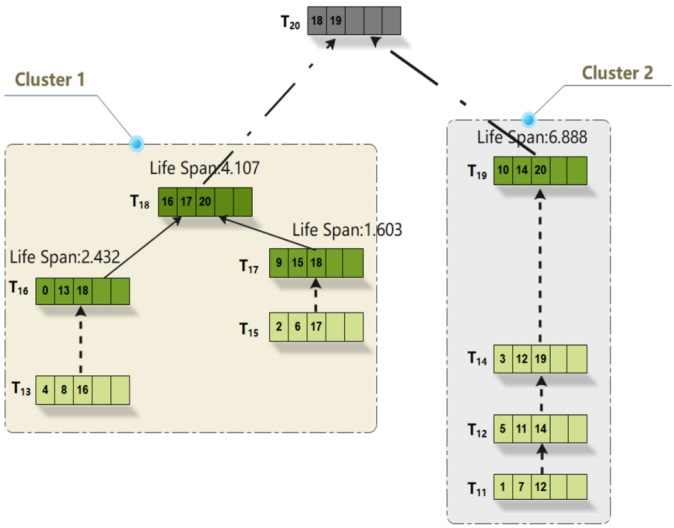
Generally, constructing a complete hierarchy via AGNES has a time complexity of $O(n^3)$ (or $O(n^2 \log\ n)$ with single link by employing efficient structures like a priority queue or a Minimum Spanning Tree) and a space complexity of $O(n^2)$. As outlined in Algorithm 2, this can be improved to $O(t^2 \log\ t)$, where $t \ll n$. Specifically, for large datasets, we have $t^2 < n$, indicating that our single-link procedure achieves significantly reduced time and space complexity, since $O(t^2) < O(n)$.

5 Automatic cluster extraction

In this section, we propose a method to automatically extract cluster information. For this purpose, a different form of dendrogram will be constructed from another output T of Algorithm 2.

Conventional techniques for cluster extraction depend on a pre-specified number of clusters; accordingly, the dendrogram will be split horizontally. Since there are n possible cluster assignments in a hierarchy, it is always troublesome work to determine the correct level when no prior knowledge is available. By contrast, the hierarchy H constructed in the compressed representation reduces such difficulty since there remain only t possibilities. However, we sometimes still expect a reliable method to automatically determine a reasonable level by which to split the dendrogram, which is also a fundamental problem in hierarchical clustering [9, 21].

The term Lifetime introduced in [18] is used as a decision basis that compares the existing time of adjacent levels from the top down. The comparison will not stop if the lifetime of



Tree No.	Status	Left Child	Right Child	Parent	Birth Time	Life Span
11	Born	1	7	12	6.972	-1
12	Grown	5	11	14	6.972	-1
13	Born	4	8	16	6.622	-1
14	Grown	3	12	19	6.972	-1
15	Born	2	6	17	5.793	-1
16	Grown	0	13	18	6.622	2.432
17	Grown	9	15	18	5.793	1.603
18	Agg	16	17	20	4.190	4.107
19	Grown	10	14	20	6.972	6.888
20	Agg	19	18	-1	0.084	-1

Fig. 4 An illustration of the extraction of steady trees on toy dataset. The first figure is the dendrogram constructed according to data structure T shown on the right side

the current level is longer than that of the lower one. However, the definition of a lifetime from a global perspective will cover the real information of a local cluster. For example, two successive levels occurring in two clusters that are far away may lead to a very short lifetime, which will obfuscate the information of local clusters of steady structures.

Consider Algorithm 2, the output H contains t levels, among which the first level corresponds to the original t-cluster partition, and each of the following levels emerges from the agglomeration of a pair of trees (in the dendrogram view) with the highest similarity among the previous level. At the same time, another data structure T is constructed to preserve information about the steady relationship. In order to overcome the mentioned limitation of global splitting a hierarchy, we propose a method that reveals 'steady trees' within a hierarchy from a local perspective. For elucidating the process, some definitions related to a tree will be introduced.

Let H[k] represents the k_{th} level of the hierarchy H, we first define a $Branch$ and a $Tree$ as:

Definition 5.1 *(Branch)* The elements in H[0] correspond to t Branches numbered from 0 to $t-1$.

Definition 5.2 *(Tree)* For any $i \in [1, t-1]$, a $Tree$ numbered as $(t-1+i)$ is generated from the agglomeration of a pair of $Trees(Branches)$ with the maximum similarity among H[i − 1].

Definition 5.3 *(life span of a Tree)* For any $Tree$ $T_i (i \geq t)$, the life span of T_i, denoted as $Ls(T_i)$, is defined as the similarity interval between the birth time and death time.

To facilitate description, we will use the notation T_i to represent a $Branch$ or a $Tree$. For cases of $i < t$, T_i is a $Branch$, otherwise a $Tree$ according to Definition 5.1 and Definition 5.2.

Note that, the definition of life span is only for a $Tree$, so we default set the life span of a $Branch$ $(T_i (i < t))$ as 0. Besides, we set the life span of the $Tree$ agglomerated at the root to -1 so that to ensure that at least two $Trees$ will be extracted in the worse case.

Motivated by the idea from [18, 21], we use the life span of a $Tree$ to describe the stability of the $Tree$ depending on the birth time and death time. The main difference is that we will introduce the status of 'Born,' 'Grown,' and 'Died' for a $Tree$ in different situations in the virtue of the definitions of $Branch$ and $Tree$. Besides, we do not rely on a global density calculation for the birth time and death time of a $Tree$.

Specifically, for any level H[k], assume that T_{t-1+k} is generated from the agglomeration of T_i and T_j. Considering the status of T_i and T_j, there are 3 different situations for T_{t-1+k}:

1) if $i < t$, and $j < t$, we call this case as a new $Tree$ is 'Born' from two $Branches$, the birth time of T_{t-1+k} is recorded as the similarity between T_i and T_j.
2) if $i < t$, and $j \geq t$, it means that T_{t-1+k} is generated from a $Tree$ (T_j) and a $Branch$ (T_i). Note that, for this case, we treat T_{t-1+k} as a $Tree$ that is 'Grown' from T_j; thus, we preserve the birth time of T_j for T_{t-1+k}. The situation that $i \geq t$ and $j < t$ is similar to the above analysis.
3) if $i \geq t$, and $j \geq t$, T_{t-1+k} is also a new $Tree$ agglomerated from two $Trees$, which simultaneously represents that T_i and T_j have 'Died.' Therefore, in addition to recording the birth time of T_{t-1+k}, we further record the

life span of $T_i(T_j)$ as the interval of the birth time of $T_i(T_j)$ and the birth time of T_{t-1+k}.

According to the above analysis, each row of T corresponds to a $Tree$ and comprises the necessary information for extracting steady $Trees$ rather than globally splitting from the hierarchy H. Particularly, the first row of T that corresponds to T_t consists of 5 different kinds of information: the number of left child, the number of right child, the number of parent, the birth time of current $Tree$, and the life span of current $Tree$.

Definition 5.4 ($steady\ Tree$) For any Tree $T_i (i \geq t)$, let $T_k (k > t)$ and $T_j (j \geq t)$ be the Parent and Sibling of T_i, respectively. We say T_i is a $steady\ Tree$ only if it satisfies: 1)$Ls(T_i) > \rho$, 2) T_j is a $Tree$, 3) $max\{Ls(T_i), Ls(T_j)\} > Ls(T_k)$.

The first condition requires T_i itself must have a life span longer than a specified low bound ρ. And if T_j is a $Branch$, T_i is treated as a $Tree$ that will grow into T_k by 'assimilating' a $Branch$ T_j, which is consistent with the previous concept of 'Grown,' while the last one compares the life span from a vertical perspective, which is exactly the process of morphologic change in Trees.

In practice, we can extract steady $Trees$ from the output T of Algorithm 2 by orderly (or randomly) traversing each $Tree$ 'Born' from two $Branches$ which remains unvisited.

Figure 4 gives an illustration for the extraction of steady $Trees$ from the dendrogram established according to T. Note that this experiment is performed on the same premise as the experiment shown in Fig. 3. That is, the same dataset and similarity matrix is used, which is also an extension of the results shown in Fig. 9b. The last 5 columns of the table shown on the right correspond to the output T, while the first two columns are manually added to facilitate a better description and understandability.

The detailed process is given in the virtue of the dendrogram shown on the left and the table on the right. Let us start with T_{13}, a new $Tree$ born from Branches T_4 and T_8. By checking its parent node, we turn to T_{16} and find it is grown from a Tree (T_{13}) and a Branch (T_0). So we directly jump to its parent T_{18}. Obviously, T_{18} is a new $Tree$ agglomerated from two Trees T_{16} and T_{17}; thus, we compare the life span of T_{18} with the life spans of T_{16} and T_{17}. We select the winner T_{18} and continue to visit the parent T_{20} which is the root with -1 as its life span. Finally, T_{18} is entirely extracted as Cluster 1. Note that the next $Tree$ to be visited is T_{11}, since it is the only $Tree$ left satisfying the condition of the 'Born' $Tree$. A similar process to the extraction of Cluster 1 can be performed to extract Cluster 2.

Finally, the first cluster consists of 6 objects, numbered 0,2,4,6,8,9, and the second one consists of objects numbered 1,3,5,7,10. In this case, the cluster structure extracted by the above procedure is completely consistent with that shown in Fig. 3c, meaning that the same clusters and NGMM are obtained.

6 Experiments

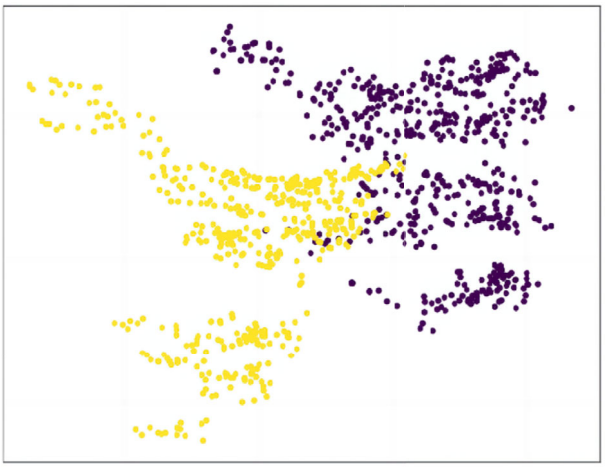
Our experiment has two main aims. First, we would like to demonstrate that our method outperforms GMM, AGNES, and other state-of-the-art clustering methods (including hierarchical, density-based methods) in terms of adaptability to different types of data by conducting experiments on a variety of standard benchmarks with quite different characteristics. Second, we want to show that our model is more scalable than AGNES and GMM in terms of data size n and dimensionality d without covariance assumptions. We, therefore, chose the dataset EMNIST Digits, consisting of 2800k 784-dimensional samples, and compared the time costs of these methods. In addition to the main purpose, we will first introduce a simple yet effective pruning trick and demonstrate the mechanism with a real-world dataset. All implementations and tests were carried out using Python 3.12.

6.1 Pruning

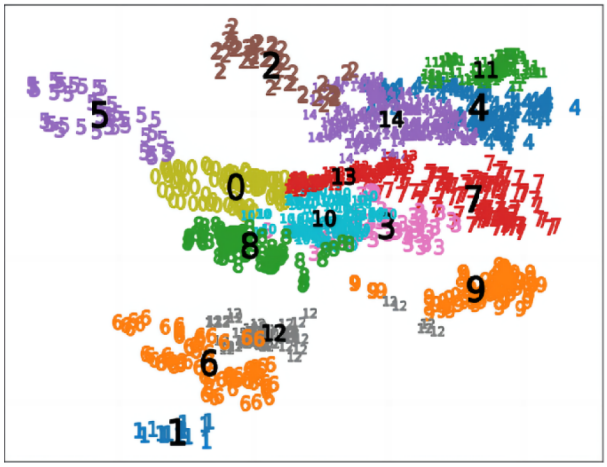
For noisy data, a simple pruning on the Gaussian capture according to the similarity matrix S may make the algorithm more robust and efficient. In detail, considering all the similarities between G_i and the rest components, if the maximum similarity is lower than a certain threshold, the component G_i is pruned from the input Gaussian capture. Formally, if $max\{S(i, j)|j \neq i, j = \{1, 2, ..., t\}\} < \delta$, the i_{th} row and column of S can be deleted.

Figure 5 shows an example of pruning based on the dendrogram of Gaussian decomposition on the real-world dataset banknote authentication, which contains two non-Gaussian clusters. Each sample of Banknote has 4 dimensions. The first sub-figure shows the point distribution of the two clusters from the Banknote using the first two dimensions. The Gaussian decomposition shown in Fig. 5b was obtained with 15 components. As illustrated in Fig. 5c, a dendrogram capturing structural information was obtained over the Gaussians in accordance with Eq. 8.

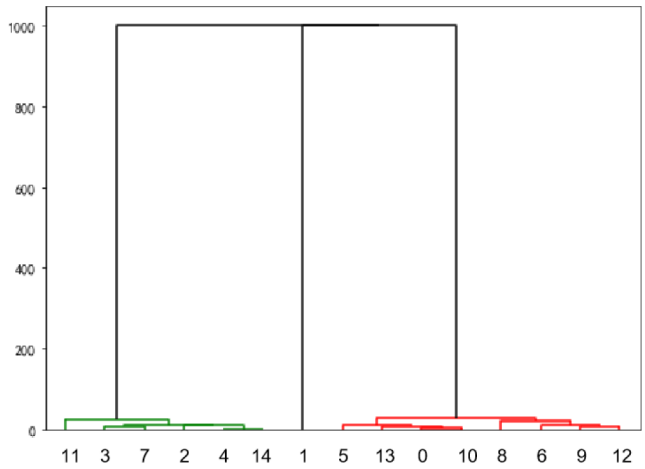
By observing the dendrogram, a 3-cluster solution stands out from all the possible solutions if we do not perform a pruning, which corresponds to the case shown in Fig. 5d. However, it is also clear from the dendrogram that Gaussian 1 can be pruned as it is a singleton cluster consisting of a few points far from other clusters (with clear lower similarity). Thus, except for determining the two-level cluster structure, the dendrogram over Gaussian decomposition can also provide us with the decision basis for pruning. Note that, this procedure may improve the general clustering performance;



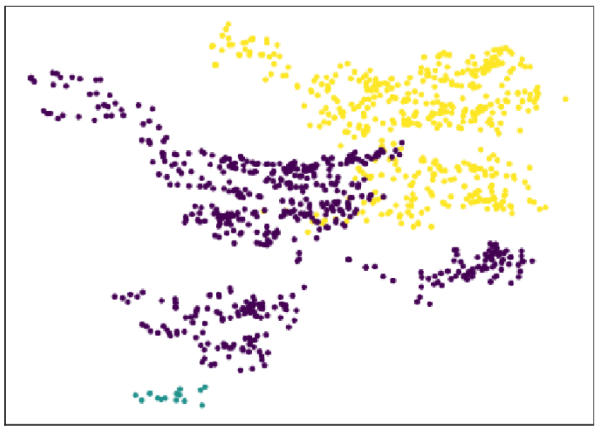
(a) Banknote-authentication consisted of 2 non-Gaussian clusters (only first two out of 4 dimensions are shown).



(b) A Gaussian decomposition consisting of 15 components.



(c) The dendrogram over Gaussian decomposition: a noticeable 3-cluster solution in NGMM emerges within the total hierarchy, where Gaussian 1 can be pruned (distant singleton cluster with low weights).



(d) Three clusters detected without pruning.

Fig. 5 An example of pruning decision on dataset Banknote authentication. Points with different colors indicate different clusters

however, it may require empirical decision based on the dendrogram.

6.2 Clustering performance comparison

We report the clustering performances of 10 algorithms on 9 datasets [11, 14] from the real world, which are selected based on characteristics including sizes (from 0.8k to 280k), dimensions (from 4 to 784), categories (from 2 to 26), and application scenarios, while according to the practical application scenarios, the datasets can be briefly classified as text categorical data (Authorship), processed image data (Banknote, Satellite Image), handwritten digits (Mf-karhunen,

Table 1 Details of datasets

Dataset	Objects	Attributes	Classes
Authorship	841	70	4
Banknote	1372	4	2
SatelliteImage	6435	36	6
Mf-karhunen	2000	64	10
Mf-factors	2000	216	10
Optdigits	5620	64	10
MiceProtein	1080	81	8
EMNIST Digits	280k	784	10
EMNIST Letters	145.6k	784	26

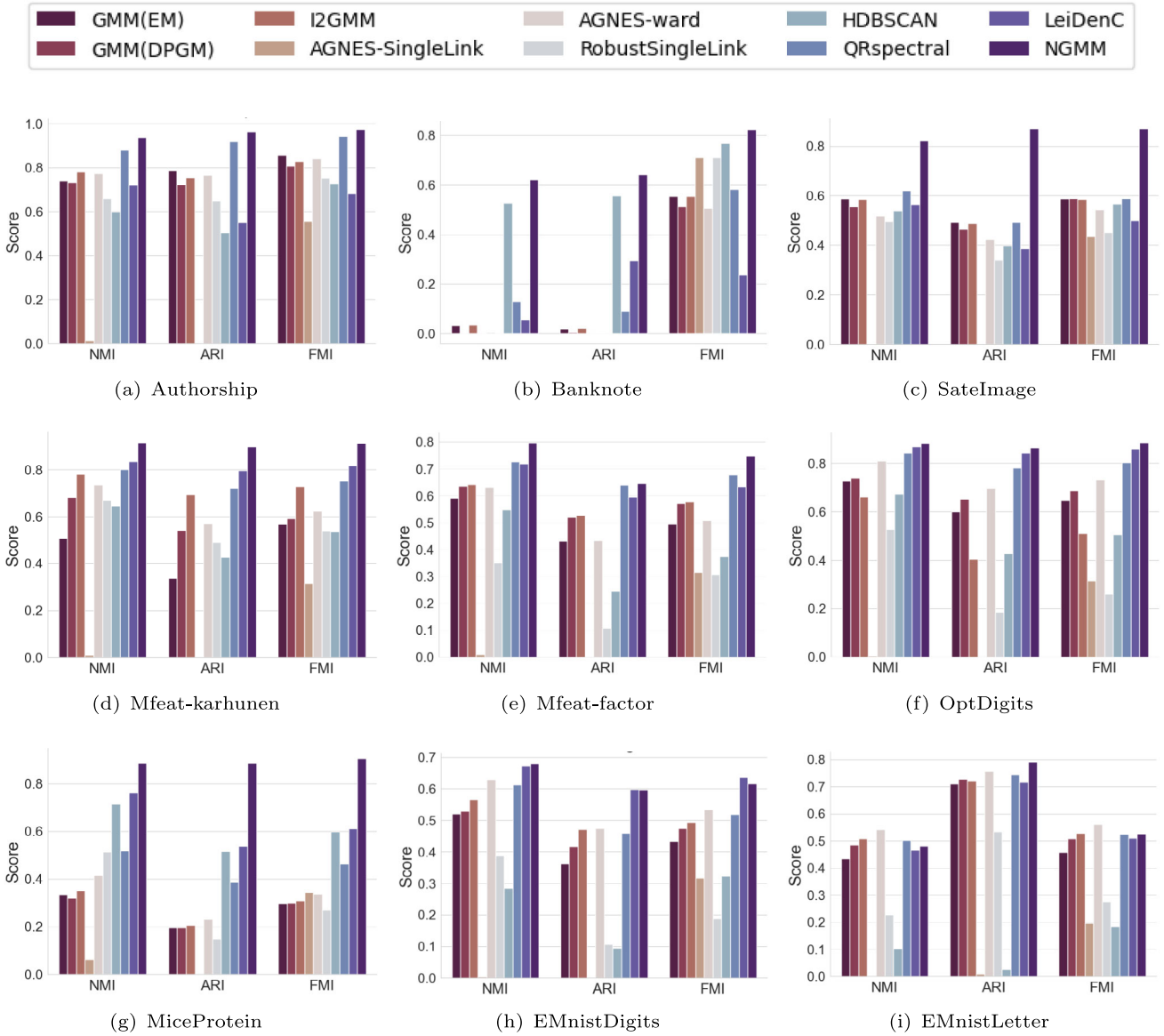


Fig. 6 Experimental results of the 10 algorithms on 9 datasets

Mf-factors, OptDigits, EMnistDigits), handwritten Letters (EMnistDigits), and protein expression data (Miceprotein). See Table 1 for more details.

The classical AGNES clustering method is used as a benchmark. In addition to the SingleLink similarity, performances by using 'Ward' [28], a more recent method, are also reported. The only used parameter for AGNES, the number of clusters, is set in accordance with the datasets.

Clustering with robust Single Link (RSL) generates Single Link connected components from different level sets as cluster trees [10]. The algorithm uses three parameters, among which the *minimal number of points* (MinPts) is prespecified to yield level sets by varying radius r from 0 to $+\infty$, while another parameter α adjusts the reachability between objects within the current level set to generate connected components. The final cluster trees are extracted as flat clusters by specifying a proper r. Following the suggestion of the original literature, we set $\alpha = \sqrt{2}$ and then search MinPts as well as a r for the best performances on all datasets.

HDBSCAN is a hierarchical algorithm working in a divisive manner according to the descending order of single linkage in the MST [8, 9]. The parameter MinPts (similar to RSL) is used to perform a nonlinear transformation of original datasets as well as define the density of objects. We report the performance of HDBSCAN(ϵ)[24], which introduces an additional parameter ϵ to avoid cases in which clusters are mistakenly separated into tiny groups. As to the parameter ϵ,

a condensed dendrogram derived from the algorithm can be referred to determine a possible range.

QRspectral [12] and Leiden clustering [37] are two recent clustering methods using adjacency (similarity) graphs. Since we use kNN graph for QRspectral to construct the similarity matrix, thus QRspectral and Leiden clustering both use parameter k to determine the neighborhood. For Leiden clustering, another parameter is needed to control dimensionality reduction.

EM and DPGM are two algorithms for estimating Gaussian mixture distribution. For EM, the main parameter is the number of components [5]. For DPGM, we apply the Dirichlet process inference algorithm [7], which can automatically activate the effective components. Besides, a greater value of the parameter concentration will force the algorithm to activate more components. Thus we set the components as two times the true number of clusters and vary the concentration value for desirable results. Besides, I2GMM, as a HGMM, is also compared. It involves two concentration parameters for the top and bottom layers of DP to be set. We mainly tune these two parameters using grid search.

Note that, for the main parameters of the above algorithms that can not be simply specified, proper searching (or grid searching) processes were conducted from varying ranges according to the parameters (and algorithms) to be determined so that we can ensure a relatively fair comparison. In addition, we use the diagonal covariance matrix to perform experiments in all cases.

The performance of the algorithms is evaluated with 3 well-known clustering evaluation indices: *Adjusted Rand Index* (ARI) [20], *Normalized Mutual-information Index* (NMI) [35], and *Fowlkes-Mallows Index* (FMI) [17]. ARI has a score from -1 to 1, where negative values are bad results, a positive value indicates better label assignments, a score close to 1 is the perfect value, and 0 stands for random assignments. The NMI has the best score equal to 1 and random assignation around 0. The FMI score ranges from 0 to 1, and a high value implies a better assignment. Note that FMI does not penalize random assignation, so it will sometimes yield lower, more cogent high scores, especially for imbalanced datasets. For such cases, the scores of ARI and AMI will close to 0.

The results of the proposed algorithm and the competitors on 9 datasets are shown in respective bar plots in Fig. 6, each of which consisted of the scores of an algorithm on 3 mentioned criteria. It can be observed that the scores of the original single-linkage method on ARI and AMI are close to 0 for almost all cases. Under these circumstances, the FMI score is no longer reliable, even though it may seem to perform better sometimes. In terms of the combined performance of agglomerative clustering methods on these datasets, the 'Ward' similarity metric has clear advantages over other metrics. In contrast, although our method also applies a procedure similar to the SingleLink aggregation, the similarity between (abstract) objects is actually quite different from the SingleLink method that measures only the distance between a pair of objects, which makes our method more stable to the impact of noise.

Table 2 Number of clusters revealed by 3 algorithms

Algorithm	DPGM	I2GMM	NGMM
Authorship	12	7	**4**
Banknote	27	12	**2**
SatelliteImage	22	13	7
Mf-karhunen	24	18	9
Mf-factors	31	14	8
Optdigits	24	16	9
MiceProtein	32	21	**8**
EMNIST Digits	28	23	14
EMNIST Letters	37	32	23
Error counts	151	70	**12**

The error count of each method is made by accumulating the difference between the predicted and true number of clusters on different datasets
Bold numbers indicate that the algorithm correctly identifies the true number of clusters

According to the performance of GMM-based methods, we can qualitatively analyze the profile of the datasets with respect to the distribution characteristics of clusters. For example, we can confirm that the Banknote is composed of non-Gaussian clusters since EM, DPMG, and I2GMM all fail in 3 criteria. Besides, the rest of the datasets should comprise both Gaussian clusters and clusters with unknown distribution since DPGM and I2GMM tend to give solutions containing more clusters than the ground truth. Overall, all these datasets can be viewed as samplings from heterogeneous mixture distributions. Thus, we can attempt to approximate the unknown distributions using equivalent NGMMs. As shown in Fig. 6, our method yields comparable or even better solutions for all the cases. Especially on Banknote, SatelliteImage, and MiceProtein, it shows considerable superiority on all the 3 criteria, while in the worst case (EMnist Digits and letters), the proposed method still gives comparable solutions to the best ones.

Besides, we also make a comparison on the consistency between the ground-truth number of clusters and the number of clusters automatically identified by NGMM, DPGM, and I2GMM. As shown in Table 2, NGMM is more consistent with the real situations than the other 2-GMM-based competitors, which tend to divide the real distribution of clusters into more fine-granularity data colonies.

6.3 Runtime comparison

To evaluate the efficiency of our method, we made a comparison of the running time between the proposed method

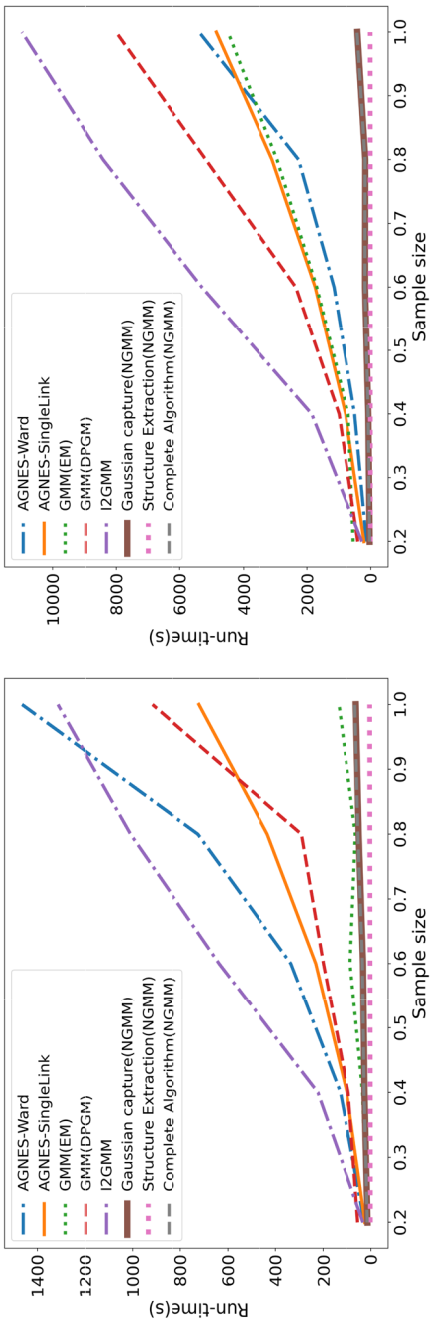
and other methods. Our method includes two main parts that may contribute to the whole running time. The first part, the estimation of a Gaussian capture, has a time complexity of $O(ctd^q n)$ where $q \in \{0, 1, 2\}$. In particular, $q = 2$ is the case of using the full covariance matrix; $q = 1$ and $q = 0$ correspond to the cases of using the diagonal and 'spherical' forms of the covariance matrix, respectively. Note that, in the experiments, the diagonal covariance matrix was used to determine a Gaussian capture. The second part corresponds to the process of hierarchy construction and structure extraction, which has a time complexity of $O(t^2)$. In addition to the overall runtime, we also report the individual runtimes of the two parts separately. For comparison, the runtimes of 3 GMM-based methods and AGNES using 2 different similarity metrics were reported.

As shown in Fig. 7, EMnist Digits, which has 280,000 samples and 784 dimensions, perform experiments comparing the runtime of different methods as sample size and dimension grow. The full dataset was used and randomly sampled to generate 4 subsets with proportion growth from 0.2 to 0.8 (in detail, 0.2, 0.4, 0.6, 0.8). The experiments include two parts: the first part used the first 100 dimensions of the various subsets of EMNIST Digits, while the second part used the subsets with all 784 dimensions.

Therefore, each part of the experiments was actually performed on 5 progressively increasing sample sets to show the increasing trends of running times of the mentioned methods with respect to the sample size n. It can be clearly observed in the two settings our method shows a far lower growth trend than the competitors, indicating better scalability in n. On the other hand, comparing the two line charts, our method also shows better scalability in d, as it exhibits less sensitivity to the growth of dimension. In addition, we can observe that the curve corresponding to the 'Gaussian capture' process gradually coincides with the curve of the 'Structure extraction' process, meaning that the time cost of our method mainly depends on the determination of a proper Gaussian capture. Overall, the result is exactly in line with the theoretical analysis of time complexity.

7 Conclusion

In this paper, we formalized the problem of clustering data from heterogeneous mixture distributions and proposed the NGMM as a solution. By decomposing unknown cluster structures into Gaussian captures and reconstructing them through hierarchical similarity analysis, NGMM can effectively approximate generalized mixture distributions without requiring prior knowledge of the underlying data distribution. Compared with traditional GMMs, our approach works under more general assumptions than Gaussian-only ones and avoids restrictive covariance settings, thus improv-



(a) The first 100 dimensions of the dataset are used (b) All the 784 dimensions are used

Fig. 7 Runtime test on samplings from EMNIST Digits evaluated with an Intel i7-10700 and 32GB RAM (The curves representing Gaussian capture basically coincide with those of the complete NGMM, indicating that the structure extraction part is negligible)

ing adaptability to non-Gaussian clusters while maintaining computational efficiency. Compared with classical hierarchical clustering methods, NGMM reduces the time complexity of hierarchy construction and shows greater robustness to noise by modeling similarities between Gaussian objects rather than raw data points.

Future work could extend the present framework in several directions. The current study employs CCR-based similarity to model the relationships among Gaussian components and a hierarchical agglomeration procedure to recover the second-level structure. Alternative approaches to similarity modeling, such as divergence- or kernel-based measures, may provide richer representations of component relationships. In addition, beyond hierarchical clustering, second-level structures could be constructed with other strategies, such as graph-based or spectral clustering methods, which may further enhance adaptability to diverse data characteristics.

Appendix A reference for Gaussian captures

Sometimes we need a reference rather than empirically specifying the value t. Therefore, we show how to determine a reference value z.

In practice, the posterior probability $p(\mathcal{M}_t|X)$ can be used to evaluate model $\mathcal{M}_t$ when varying the value of t. Another important factor concerning t that should be taken into account is the efficiency of the algorithm. A large t will bring about excessive time cost to our method. This makes us consider imposing a penalty on large t and data size n. According to the above analysis, we use the following function w.r.t. t to evaluate $\mathcal{M}_t$:

$$J(t) = p(\mathcal{M}_t|X) - g(t)ln(n) \tag{A1}$$

where $g : t \to R^+$ is a function of t, which acts as a penalty term rejecting candidates consisted of excessive components to meet the constraint that $t \ll n$.

According to the Bayesian formula, the posterior of $\mathcal{M}_t$ bears the following property:

$$p(\mathcal{M}_t|X) \propto p(X|\mathcal{M}_t)p(\mathcal{M}_t) \tag{A2}$$

where $p(\mathcal{M}_t)$ is the prior probability of $\mathcal{M}_t$, and $p(X|\mathcal{M}_t)$ is the likelihood function.

Let G_j be the j_{th} Gaussian component of $\mathcal{M}_t$, by assuming that all candidate models have the identical prior probability, we have:

$$p(\mathcal{M}_t|X) \propto p(X|\mathcal{M}_t) = \prod_{i=1}^{n}\sum_{j=1}^{t}\alpha_j \cdot p(x_i|G_j) \tag{A3}$$

where α_j is the mixture coefficient, and $p(x_i|G_j)$ is probability of x_i w.r.t. G_j:

$$p(x_i|G_j) = \frac{1}{\sqrt{2\pi}^{\frac{n}{2}}|\Sigma_j|^{\frac{1}{2}}} \cdot e^{(x_i-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)} \tag{A4}$$

where μ_j, Σ_j are the mean and covariance matrix of G_j, respectively.

For the convenience of calculation, we use the log-likelihood and take $g(t) = t$. The evaluation function in Eq. A1 is rewritten as:

$$\begin{aligned} J(t) &= -log(\ p(X|\mathcal{M}_t)) + g(t)ln(n) \\ &= -\sum_{i=1}^{n} log \sum_{j=1}^{t}\alpha_j \cdot p(x_i|G_j) + t \cdot ln(n) \end{aligned} \tag{A5}$$

Note that, when we set $g(t)$ equal to $t/2$, $J(t)$ is exactly the Bayesian information criterion. Now a higher score of $J(t)$ implies t is approaching z. Thus by minimizing $J(t)$, we give the reference value of z:

$$z = argmin_t J(t) \tag{A6}$$

Figure 8 shows an example on the artificial dataset M2 to illustrate the determination of a reference value z. Different types of the covariance matrix are tested in the experiments. In detail, as for the case of using full covariance, we get the minimum score when the number of components is equal to 6. It means that $z = 6$ can be taken as the reference value for determining a Gaussian capture which is exactly the result shown in Fig. 9a when $t = z$. Similarly, for the case of diagonal covariance, we have $z = 11$, and the corresponding Gaussian capture is shown in Fig. 9b. For the final case in which that 'spherical' covariance matrix is used, the reference value for z is 16. Note that, it is a little different for the result shown in Fig. 9c compared with the above two cases since we take $t = 20(t > z)$ to obtain a Gaussian capture. This result is consistent with our analysis that $\mathcal{M}_t$ is Gaussian captures when $t > z$.

Figure 9 shows examples of 3 different types of Gaussian capture obtained by using different forms of the covariance matrix. We can observe that on artificial datasets M2 and M3, Gaussian captures can be determined by using incomplete forms of the covariance matrix. In particular, the diagonal covariance matrix captures the ellipsoid or spherical Gaussian decompositions (see Fig. 9b,e), while the 'spherical' covariance matrix, i.e., the form of the scalar matrix, captures only the spherical Gaussian decompositions (see Fig. 9c,f).

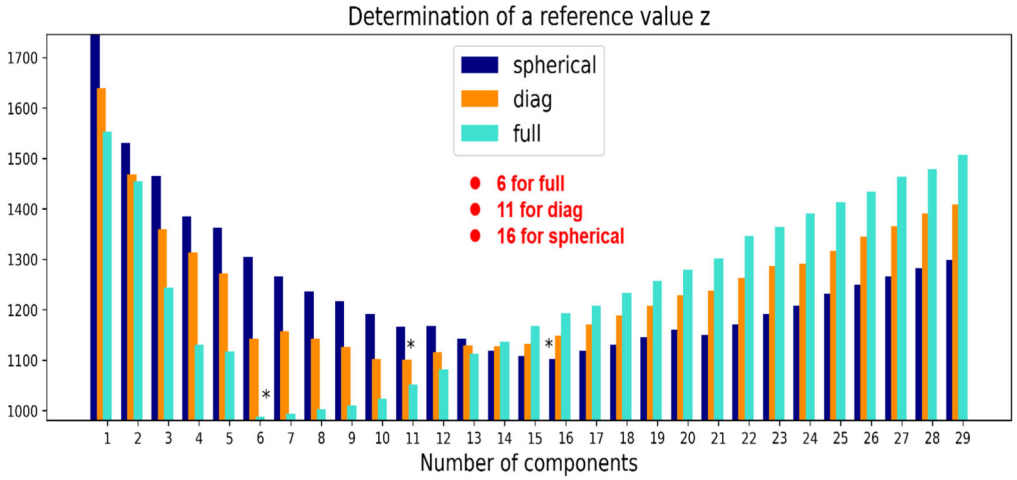
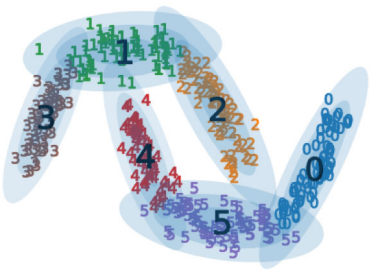


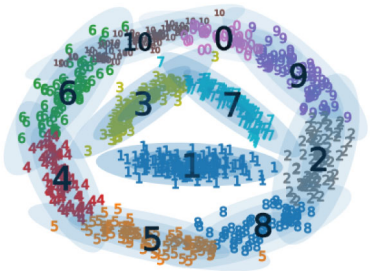
Fig. 8 An example of determining a reference z for Gaussian capture with different types of covariance matrix on artificial dataset M2



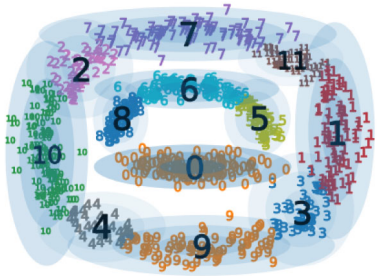
(a) Gaussian capture with 6 components using full covariance matrix on M2

(b) Ellipsoid Gaussian capture with 11 components on M2

(c) Spherical Gaussian capture with 20 components on M2



(d) Gaussian capture with 11 components using full covariance matrix on M3



(e) Ellipsoid Gaussian capture with 12 components on M3

(f) Spherical Gaussian capture with 25 components on M3

Fig. 9 Gaussian captures on M2 and M3 when different Gaussian assumptions are made (by applying spherical, diag, and full covariance matrices)

Author Contributions Fei Wang helped in conceptualization, methodology, software, writing—reviewing, and editing. Le Li contributed to data curation, writing original draft preparation. Pasi Fränti helped in guidance, revisions. All authors read and approved the final manuscript.

Funding This work was supported by the Finnish Doctoral Program Network in Artificial Intelligence, AI-DOC (decision number VN/3137/2024-OKM-6).

Data Availability The dataset we used is publicly available.

Code availability The code is available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

Ethics approval and consent to participate Not applicable.

Consent for publication All authors of this manuscript consent to its publication.

References

1. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. Tech. rep (2006)
2. Athey, TL., Liu, T., Pedigo, B.D., et al: AutoGMM: Automatic and Hierarchical Gaussian Mixture Modeling in Python. Preprint at arXiv:1909.02688 (2019)
3. Bahrololum, M., Khaleghi, M.: Anomaly intrusion detection system using Gaussian mixture model. Int. J. Comp. Sci Network Secur. **8**(8), 264–271 (2008)
4. Berkhin, P.: A survey of clustering data mining techniques. In: Grouping multidimensional data: Recent advances in clustering, pp. 25–71. Springer, Berlin Heidelberg (2006)
5. Bilmes, J.A.: A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Int. Comp. Sci. Inst. **4**(510), 126 (1998)
6. Bishop, C.M.: Pattern recognition and machine learning. Springer (2006)
7. Blei, D.M., Jordan, M.I.: Variational inference for Dirichlet process mixture models Mean field variational inference. Bayesian Anal. **1**, 121–143 (2006)
8. Campello, R.J., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pacific-Asia conference on knowledge discovery and data mining, pp. 160–172. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-37456-2_14
9. Campello, R.J., Moulavi, D., Zimek, A., et al.: Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM Trans. Knowl. Discov. Data **10**(1), 1–51 (2015). https://doi.org/10.1145/2733381
10. Chaudhuri, K., Dasgupta, S.: Rates of convergence for the cluster tree. In: NIPS, pp. 343–351. (2010)
11. Cohen, G., Afshar, S., Tapson, J., et al.: Emnist: Extending mnist to handwritten letters. In: 2017 international joint conference on neural networks (IJCNN), pp. 2921–2926. (2017)
12. Damle, A., Minden, V., Ying, L.: Simple, direct and efficient multi-way spectral clustering. Inf. Inference A J. IMA **8**(1), 181–203 (2019). https://doi.org/10.1093/imaiai/iay008
13. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. J. Roy. Stat. Soc.: Ser. B (Methodol.) **39**(1), 1–38 (1977)
14. Dua, D., Graff, C.: UCI machine learning repository. http://archive.ics.uci.edu/ml (2017)
15. Eckart, B., Kim, K., Troccoli, A., et al.: Accelerated generative models for 3D point cloud data. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5497–5505. (2016)
16. Eckart, B., Kim, K., Kautz, J.: HGMR: Hierarchical gaussian mixtures for adaptive 3D registration. In: proceedings of the European conference on computer vision (ECCV), pp. 705–721. (2018)
17. Fowlkes, E.B., Mallows, C.L.: A method for comparing two hierarchical clusterings a method for comparing two hierarchical clusterings. J. Am. Stat. Assoc. **78**(383), 553–569 (1983)
18. Fred, A.L., Jain, A.K.: Combining multiple clusterings using evidence accumulation. IEEE Trans. Pattern Anal. Mach. Intell. **27**(6), 835–850 (2005). https://doi.org/10.1109/TPAMI.2005.113
19. Goldberger, J., Roweis, S.: Hierarchical clustering of a mixture model. In: Advances in neural information processing systems, vol. 17, (2004)
20. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**(1), 193–218 (1985). https://doi.org/10.1007/BF01908075
21. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Inc, (1988)
22. Kass, R.E., Wasserman, L.: A reference Bayesian test for nested hypotheses and its relationship to the Schwarz criterion. J. Am. Stat. Assoc. **90**(431), 928–934 (1995). https://doi.org/10.1080/01621459.1995.10476592
23. Malzer, C., Baum, M.: A hybrid approach to hierarchical density-based cluster selection. In: 2020 IEEE international conference on multisensor fusion and integration for intelligent systems (MFI), pp. 223–228. (2020). https://doi.org/10.1109/MFI49285.2020.9235263
24. McInnes, L., Healy, J., Astels, S.: hdbscan: hierarchical density based clustering. J. Open Source Softw. **2**(11), 205 (2017). https://doi.org/10.21105/joss.00205
25. McLachlan, G.J., Peel, D.: Finite mixture models. Wiley (2000)
26. Murtagh, F., Contreras, P.: Algorithms for hierarchical clustering: An overview. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **2**(1), 86–97 (2012). https://doi.org/10.1002/widm.53
27. Murtagh, F., Contreras, P.: Algorithms for hierarchical clustering: An overview, II. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **7**(6), e1219 (2017)
28. Murtagh, F., Legendre, P.: Ward's hierarchical clustering method: clustering criterion and agglomerative algorithm. J. Classif. **31**(3), 274–295 (2014)
29. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science **344**(6191), 1492–1496 (2014). https://doi.org/10.1126/science.1242072
30. Sadrishojaei, M.: Energy-efficient routing for internet of things using combination of meta-heuristic algorithms in viral pandemics. IETE Journal of Research pp. 1–11. (2025). https://doi.org/10.1080/03772063.2025.2487936
31. Sadrishojaei, M., Kazemian, F.: Clustered routing scheme in iot during covid-19 pandemic using hybrid black widow optimization and harmony search algorithm. In: Operations Research Forum, p. 47. Springer, Cham (2024). https://doi.org/10.1007/s43069-024-00331-x
32. Sadrishojaei, M., Kazemian, F.: New routing method in flying ad hoc networks based on squirrel search algorithm. Int. J. Comput. Appl. **47**(6), 518–531 (2025). https://doi.org/10.1080/1206212X.2025.2502778
33. Sibson, R.: Slink: an optimally efficient algorithm for the single-link cluster method. Comput. J. **16**(1), 30–34 (1973). https://doi.org/10.1093/comjnl/16.1.30

34. Sokal, R.R., Michener, C.D.: A statistical method for evaluating systematic relationships. Univ. Kansas Sci. Bulletin **38**, 1409–1438 (1958)
35. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2002)
36. Stuetzle, W., Nugent, R.: A generalized single linkage method for estimating the cluster tree of a density. J. Comput. Graph. Stat. **19**(2), 397–418 (2010). https://doi.org/10.1198/jcgs.2009.07049
37. Traag, V.A., Waltman, L., Van Eck, N.J.: From louvain to leiden: guaranteeing well-connected communities. Sci. Rep. **9**(1), 5233 (2019). https://doi.org/10.1038/s41598-019-41695-z
38. Wang, F., Li, L., Liu, Z.: Stratification-based semi-supervised clustering algorithm for arbitrary shaped datasets. Inf. Sci. **639**, 119004 (2023). https://doi.org/10.1016/j.ins.2023.119004
39. Ward, J.H.: Hierarchical grouping to optimize an objective function. J. Am. Stat. Assoc. **58**(301), 236–244 (1963). https://doi.org/10.1080/01621459.1963.10500845
40. Xie, W.B., Liu, Z., Das, D., et al.: Scalable clustering by aggregating representatives in hierarchical groups. Pattern Recogn. **140**, 109694 (2023). https://doi.org/10.1016/j.patcog.2022.109230