Model-based Clustering with HDBSCAN*

Michael Strobl (✉)[1], Jörg Sander[1], Ricardo J. G. B. Campello[2], and Osmar Zaïane[1]

[1] University of Alberta, Edmonton, Canada
{mstrobl,jsander,zaiane}@ualberta.ca
[2] University of Newcastle, Callaghan, Australia
ricardo.campello@newcastle.edu.au

Abstract. We propose an efficient model-based clustering approach for creating Gaussian Mixture Models from finite datasets. Models are extracted from HDBSCAN* hierarchies using the Classification Likelihood and the Expectation Maximization algorithm. Prior knowledge of the number of components of the model, corresponding to the number of clusters, is not necessary and can be determined dynamically. Due to relatively small hierarchies created by HDBSCAN* compared to previous approaches, this can be done efficiently. The lower the number of objects in a dataset, the more difficult it is to accurately estimate the number of parameters of a fully unrestricted Gaussian Mixture Model. Therefore, more parsimonious models can be created by our algorithm, if necessary. The user has a choice of two information criteria for model selection, as well as a likelihood test using unseen data, in order to select the best-fitting model. We compare our approach to two baselines and show its superiority in two settings: recovering the original data-generating distribution and partitioning the data correctly. Furthermore, we show that our approach is robust to its hyperparameter settings.[1]

Keywords: Hierarchical Clustering · Expectation Maximization · Model Selection

1 Introduction

Model-based clustering is a popular tool for unsupervised data analysis due to its powerful and compact representation of each group in a dataset. The Expectation Maximization (EM) [9] algorithm uses Maximum Likelihood (ML) estimation for fitting Multivariate Gaussian models to data quickly and accurately. These models are also known as Gaussian Mixture Models (GMM).

However, there are a number of points to consider when using EM. First, the number of clusters K (corresponding to the number of components in the final model) has to be known in advance since EM tries to fit the parameters of a GMM with K components to a dataset. Due to EM's ML parameter estimation based on a finite number of observations, setting this parameter correctly is

[1] Data and code are publicly available at: https://github.com/mjstrobl/HCEM

crucial as more components lead to a higher number of parameters and a higher loglikelihood of the data given the model. This is also known as model overfitting (i.e. high divergence to the original data-generating distribution). As a result, the loglikelihood of the data given a GMM cannot be used to select a model among a set of differently parameterized models with varied K. In addition, the covariance matrix parametrization of each component can be either fully open or partially restricted. This leads to more or less parsimonious models with Gaussians of equal or variable volume, shape, or orientation. Specifically for small datasets, it can be beneficial to choose a more restricted model. Second, EM has to be initialized with K objects, e.g. from the dataset, as seeds for each of the K initial clusters, in order to fit a model with K components. However, EM is very sensitive to the choice of those K seed objects. Therefore, several (typically non-deterministic) initialization strategies exist, e.g. choose K random seed objects. Depending on the hyperparameter setting, they are run multiple times in order to find the best initialization parameters.

Another popular clustering paradigm is density-based clustering, as introduced by the DBSCAN [10] algorithm and more recently extended by its hierarchical version HDBSCAN* [4]. DBSCAN's clustering model is deterministic, relatively fast to compute, and less strict than GMMs. It allows clusters of arbitrary shapes and the number of clusters does not have to be known in advance. Noise in the data can be detected, whereas GMMs have to model noise separately. However, if the assumption of Gaussian distributed clusters applies (even approximately), different Gaussian clusters in a dataset may be merged into a single cluster by DBSCAN, if they are overlapping, due to the difficulty of setting an appropriate global density-threshold. HDBSCAN*, on the other hand, uses a cluster stability measure to extract clusters from the cluster hierarchy. Cluster Stability may also favor the selection of a cluster (node in the hierarchy) that represents the result of merging child nodes representing two clusters that are generated by different, possibly overlapping, Gaussian model components. HDBSCAN* hierarchies, consisting of clusters with varying densities, often contain all clusters corresponding to each component of a generating GMM (even though these nodes may not be selected by Cluster Stability). Such hierarchies can therefore be used to find a GMM that fits the data well, when a more suitable cluster extraction strategy is used.

In this paper, we are proposing a clustering framework for model-based clustering that is able to accurately recover the original data-generating distribution for GMMs. Furthermore, it can automatically select the number of components K of the model. It is based on the HDBSCAN* hierarchy, which provides a compact tree of clusters from which models of different sizes can be extracted and evaluated efficiently.

The remainder of this article is structured as follows: Section 2 introduces GMMs and EM as well as HDBSCAN* in more detail. Our framework is introduced in Section 3 with experiments in Section 4 to support its strengths compared to two baselines. The article concludes in Section 5.

2 Background and Related Work

In this section we provide some background and related work on the popular model-based clustering paradigm, which our approach follows. Furthermore, we describe the HDBSCAN* algorithm for hierarchical density-based clustering, from which models can be extracted. For the rest of this article we assume our model is a mixture of multi-dimensional Gaussians, referred to as GMM.

2.1 Model-based clustering

Model-based clustering aims to estimate the parameters of a GMM using the EM technique [9]. The goal of EM is to fit a GMM to a dataset using an iterative procedure with the loglikelihood as the optimization goal.

The loglikelihood for a dataset $\boldsymbol{X} = (x_1, ..., x_n)$ with n observations (assumed to be i.i.d.), $x_i \in \mathbb{R}^d$ with $1 \leq i \leq n$ and $d \in \mathbb{N}_+$ is defined as

$$L(\boldsymbol{X}, \Theta_K) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} p_k f(x_i | a_k) \right] \tag{1}$$

where K is the number of components (one corresponding to each cluster) and $\Theta_K = \{p_1, ..., p_{K-1}, a_1, ..., a_K\}$, with $0 < p_k < 1$ and $\sum_{k=1}^{K} p_k = 1$. Each component has a mixing proportion p_k and a parameter vector $a_k = (\mu_k, \Sigma_k)$, with μ_k as the mean and Σ_k as the covariance matrix. $f(x_i|a_k)$ is the d-dimensional Gaussian density of the component k for object x_i.

For clustering, Equation 1 is usually maximized using the EM algorithm [16], which estimates the conditional probabilities for all x_i in the Expectation (E) step and updates the model parameters in the Maximization (M) step iteratively.

Another variant of the EM algorithm for clustering is the Classification EM algorithm (CEM) [6], which aims to optimize the Classification Likelihood (CL), which is defined as

$$CL(\boldsymbol{X}, \Theta_K) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \log(p_k f(x_i | a_k)) \tag{2}$$

where C_k is the set of objects x_i belonging to cluster k.

CEM does not compute a soft classification with every x_i being a member of each cluster to some extent, instead cluster labels are assigned. Therefore, the contribution of each x_i to $CL(\boldsymbol{X}, \Theta_K)$ is solely based on the parameter vector $a_k = (\mu_k, \Sigma_k)$ and the mixing proportion p_k of the cluster C_k to which x_i belongs. Biernacki and Govaert [3] showed that the CL can be seen as a penalized version of Equation 1, favouring models with well-separated mixture components.

Unrestricted GMMs can be highly over-parameterized or, especially in high dimensional spaces, it may be difficult to estimate all parameters accurately. Therefore Celeux and Govaert developed an approach [7] to restrict the number of parameters in a GMM using the eigenvalue decomposition of Σ_k:

$$\varSigma_k = \lambda_k D_k A_k D_k'$$

where $\lambda_k = |\varSigma_k|^{1/d}$, D_k is the matrix of eigenvectors and A_k is a diagonal matrix with the normalized eigenvalues of $\varSigma_k$ in decreasing order on the diagonal. Different combinations of these parameters result in 14 models with clusters of variable or equal volumes, shapes and orientations.

Three models, resulting in independent parameter estimations for all components, allow us to extract GMMs from clustering hierarchies efficiently[3]:

– VVV ($\varSigma_k = \lambda_k D_k A_k D_k'$): Fully unrestricted; Volume, Shape and orientation of all components are variable.
– VVI ($\varSigma_k = \lambda_k A_k$): Clusters are aligned to coordinate axis.
– VII ($\varSigma_k = \lambda_k I$): Only volume is variable, components are spherical.

2.2 Traditional Initialization

The most common initialization strategy for EM is starting with K random observations from the dataset and run EM until convergence. However, this can result in a suboptimal partition, if these K random observations are too different from the real cluster centers. In order to avoid this problem, EM can be run multiple times with random initialization and the model converging to the highest likelihood is selected. A similar approach is to run EM multiple times with random initialization for only a few iterations and continue iterating with the most promising model.

If models of different sizes (numbers of components K) should be created, from which one of them can be later selected, they have to be created independently. The procedure is simply repeated as many times as necessary, for each $K \in \mathcal{K} = [K_l, K_u]$ with K_l and K_u as the lower and upper bound of K, typically provided by the user.

2.3 Hierarchical Initialization

The main disadvantage of traditional initialization strategies is that if the number of clusters is unknown, multiple solutions (one for each $K \in \mathcal{K}$) have to be computed independently. Similarities between initializations of size K and $K-1$ are not taken into account. This can be prohibitively expensive if solutions for a large $|\mathcal{K}|$ have to be evaluated. Extracting solutions from a clustering hierarchy can be advantageous in this case, because new solutions with $K-1$ clusters can be computed through merging siblings in the hierarchy quickly, in the case of an agglomerative approach. In order to determine which set of siblings should be merged, for example, the sum-of-squares criterion can be used to select siblings that would result in the lowest possible variance increase of the model, if merged.

[3] All other models contain parameters that are equal among all components and therefore have to be jointly re-estimated if components of child nodes are replaced with parent node components. More information on this is explained below in Section 3.2.

Fraley [11] describes an agglomerative hierarchical clustering algorithm based on GMMs. It starts with each observation representing its own cluster. As criterion for merging clusters, the CL is used at each stage, i.e. all pairs of clusters at the current stage are considered, but only one pair, that results in a model with the highest CL among all possible merges, is actually merged. For the models VVV, VVI and VII, it is not necessary to recompute all model parameters at each stage since each x_i contributes to the model's CL only through the parameters of the model component it belongs to. Therefore, only the new parent's parameters have to be computed. For models other than these, some parameters have to be recomputed for all components, e.g. if the volume of all components must be equal according to the chosen parametrization.

Since agglomerative hierarchical clustering algorithms start from singleton clusters, the hierarchy can take up a large amount of memory space. If it is not possible to compute all ML parameters (especially at the beginning with singleton clusters), the sum-of-squares criterion is used until clusters are large enough. While the hierarchy is built up, models of different sizes can be extracted. In order to extract only a limited number of models, a range of clustering sizes $\mathcal{K}$ can be provided (see [12]). Once there are $max(\mathcal{K})$ clusters at the top of the hierarchy (while building it), a model of size $max(\mathcal{K})$ is extracted. Clusters are continuously merged and a new model is extracted after each merge. Those models can be re-parameterized with EM and evaluated using a model selection method.

2.4 Model Selection

There are two main problems that model selection methods aim to solve: How many components the final model should have and how the covariance matrix Σ_k should be parameterized. In model-based clustering, information criteria are used to solve these problems. Their main goal is to find a trade-off between the best model-fit (high loglikelihood) and the least number of parameters in the model. The following are commonly used criteria, e.g. used in [2]:

– Bayesian Information Criterion (BIC) [19]:

$$BIC = L(\boldsymbol{X}|\hat{\Theta}_K) - \frac{\nu_K}{2}\log n$$

ν_K corresponds to the number of free parameters in the model, which depends on the number of components in the model and its parametrization.

– Integrated Complete Likelihood (ICL) [1]:

$$ICL = CL(\boldsymbol{X}|\hat{\Theta}_K) - \frac{\nu_K}{2}\log n$$

ICL penalizes models with less well-separated components more than BIC.

– Normalized Entropy Criterion (NEC) [8]:

$$NEC = \frac{E(\boldsymbol{X}|\hat{\Theta}_K)}{L(\boldsymbol{X}|\hat{\Theta}_K) - L(\boldsymbol{X}|\hat{\Theta}_1)}, \qquad E(\boldsymbol{X}, \hat{\Theta}_K) = -\sum_{k=1}^{K}\sum_{i=1}^{n} \hat{t}_{ik} \log \hat{t}_{ik} \geq 0$$

where $\hat{t}_{ik}$ is the conditional probability that x_i belongs to C_k.
The NEC favours low entropy models with well-separated components over models with more components and worse separation.

All information criteria have in common that they make models with different parametrizations and numbers of components comparable.

2.5 HDBSCAN*

HDBSCAN* [4] is a hierarchical clustering algorithm based on DBSCAN*, a revised version of DBSCAN [10], which is a well-known density-based clustering algorithm. DBSCAN* receives two input values set by the user: a density-threshold ε and the value m_{pts}, which acts as a smoothing factor for the density-estimates of each object. The original DBSCAN introduced the notion of *core objects*, which are objects with at least m_{pts} neighbours within the distance ε. These neighbours form an object's ε-neighbourhood N_ε. A cluster C is defined as a non-empty maximal subset of the input dataset, such that every pair of objects in C is density-connected. Two core objects x_p and x_q are density-connected if they are directly or transitively ε-reachable; directly ε-reachable means $x_p \in N_\varepsilon(x_q)$ and $x_q \in N_\varepsilon(x_p)$.

In case of DBSCAN*, the global density-threshold ε is difficult to set if there are clusters of varying densities, because each cluster must exceed density-level ε. HDBSCAN* solves this problem by creating a hierarchy of clusters of different densities. In order to restrict the hierarchy's depth, the optional parameter m_{cl} was introduced as a minimum number of objects within a cluster. In [4], it is suggested to set $m_{cl} = m_{pts}$ and leave m_{pts} as the only user-defined parameter, which is assumed in the remainder of this article.

The HDBSCAN* hierarchy is created by the following steps:

1. Building the Minimum Spanning Tree (MST) of the Mutual Reachability Graph, which consists of all objects as vertices and the mutual reachability distances for each pair of objects as edge weight. The mutual reachability distance is defined as
$d_{mreach}(x_p, x_q) = max\{d_{core}(x_p), d_{core}(x_q), d(x_p, x_q)\}$
where $d_{core}(x)$ is the distance of object x to its m_{pts} nearest neighbours and $d(x_p, x_q)$ is the distance between x_p and x_q.
2. The whole dataset is contained in the cluster which forms the root of the hierarchy. Subsequent clusters and noise are created by removing edges from the MST in decreasing order of weight. Removing an edge results in either two new clusters with more than m_{cl} objects each, a shrank cluster with at least m_{cl} objects and noise, or noise only if the cluster disappears.
3. From the full clustering hierarchy containing all splits, a compact hierarchy is created containing only the most important levels, namely when clusters appear the first time or completely disappear.

As a special case, setting $m_{pts} = 1$ results in a single-linkage hierarchy.

Due to the different density-thresholds represented by the hierarchy (the closer to the root the less dense the density threshold is), the user does not need to provide the parameter ε anymore. The original DBSCAN algorithm produced flat partitions whose clusters are maximal sets of connected points with a point density in their neighborhood above a given density threshold, set by the parameter ε. The result corresponds to extracting clusters from the HDBSCAN* hierarchy at the density level ε, representing a horizontal cut through the hierarchy. In order to extract clusters of variable densities (different levels) from such a hierarchy, the measure of Stability was introduced. The Stability S of a cluster measures how stable it is in the hierarchy. This is dependent on the density-range in which the cluster exists, i.e. minimum density value at which the cluster starts to exist and maximum density level at which the cluster is either split or disappears, as well as how many objects the cluster contains compared to its descendants and parent cluster. Stability values can be computed while creating the hierarchy.

The algorithm to extract a flat partition from such a hierarchy is an iterative procedure and starts at the leaf level as the current clustering solution. For every parent-descendants pair in the current solution: If a parent C_p is more stable than its descendants, say C_{d_1} and C_{d_2}, i.e. $S_{C_p} > S_{C_{d_1}} + S_{C_{d_2}}$, C_{d_1} and C_{d_2} are replaced by C_p in the current solution. Otherwise the parent of C_p is assigned to C_{d_1} and C_{d_2} as new parent cluster, basically ignoring C_p from now on. This procedure is repeated until the root of the hierarchy is the parent of all clusters in the current solution.

Extracting a clustering solution this way is efficient since at every stage local decisions are made, i.e. comparing only parents and descendants (and no other clusters) is sufficient to obtain a globally optimal solution regarding S, returning the most stable set of clusters.

3 Method

In this section we present how to create a GMM from HDBSCAN* hierarchies. To account for the variations in the cluster tree when choosing different values of m_{pts}, we use multiple hierarchies and choose the best partition according to the CL. Neto et al. [17] show how over a hundred hierarchies (i.e., different values of m_{pts}) can be efficiently computed with the cost of about 2 HDBSCAN* runs.

3.1 Classification Likelihood Criterion

While the original HDBSCAN* Stability measure works well for density-based clustering, it may not be suitable for a dataset following a GMM. Different clusters with a relatively dense overlap could be considered as a very stable single cluster since it could appear as such early in the hierarchy and be separated very late. However, extracting a clustering solution based on Stability is efficient due to local decision making. Through using the models VVV, VVI, and VII, and CL as optimization criterion, it is sufficient to make local decisions as well, while still extracting the best model (according to CL) from an HDBSCAN* hierarchy, when Gaussian distributed clusters are assumed. In [3], GMMs with different settings of K are fitted to a dataset and a CL-like criterion is used to compare these models. The authors argue that the CL works for finding the correct number of mixture components K, whereas the loglikelihood L in Equation 1 overestimates K. We argue that each mixture component is represented as separate cluster in the HDBSCAN* hierarchy and that the CL (instead of Stability) can be used to extract those clusters.

3.2 Creating GMMs from HDBSCAN* hierarchies

The high-level steps of our algorithm for clustering dataset $\boldsymbol{X}$ are the following:

1. Create HDBSCAN* hierarchies for $\boldsymbol{X}$, one for each m_{pts}-value.
2. For each hierarchy:
   (a) Create an initial GMM with one component per leaf, independently estimated.
   (b) Assign each noise object to the leaf with the highest loglikelihood according to the initial GMM and re-estimate the parameters of all components, representing a candidate model.
   (c) After estimating the Gaussian parameters of all clusters in the hierarchy, the optimal GMM (according to CL and given the hierarchy) for each model parametrization (VVV, VVI and VII) is extracted. This is done through an iterative procedure similar to using the Stability in [4].
3. Select the best model among all models and create a flat partition according to this model.

In the following, we provide a detailed description of these steps.

1. Create HDBSCAN* hierarchy. We run the original HDBSCAN* algorithm on a dataset $\boldsymbol{X}$ in order to create a compact hierarchy for each m_{pts}-value.

(a) Create initial GMM. HDBSCAN* creates a hierarchy of clusters top-down. Whenever clusters are split, noise objects may be created. Our algorithm starts by creating GMMs bottom-up, which makes it necessary to assign all noise objects to leaf clusters in the hierarchy, since the union of all leaf clusters is not necessarily equal to $\boldsymbol{X}$. For each leaf, the parameters of one Gaussian with an unrestricted Σ_k (model VVV) are estimated. If it is not possible to estimate a non-singular unrestricted Σ_k due to data sparsity, more restricted models are used, VVI or VII. Leaves where this is not possible are removed from the set of leaves including its sibling and the parent cluster is added instead. Parameter estimation is then tried again for this node. The initial GMM is defined by the set of all estimated components, one for each leaf.

(b) Assign noise objects. All noise objects are assigned to the model component l with $l = \text{argmax}_{k=1,\ldots,K}\, p_k f(x_i|a_k)$. After this step, it is possible to estimate parameters more accurately since all $x_i \in \boldsymbol{X}$ are assigned to clusters now, and, therefore, the parameters of all components are re-estimated.

(c) Create candidate GMMs. The initial GMM represents a clustering solution with the maximum possible K, given the clusters in the HDBSCAN* cluster tree. New models of smaller sizes and parametrizations (VVV, VVI or VII) can be created iteratively by merging or keeping siblings according to the best CL of the resulting model. This works the same way as done using the Stability as described in Section 2.5. The only parameters that have to be estimated are the mean μ_k and covariance matrix Σ_k of the parent, since the parameters of the descendants are estimated in the previous iteration or step (b). In addition, since we are using the VVV, VVI and VII parameterizations, parameters of other components in the current model are independent and therefore do not have to be re-estimated. Decisions are made locally using the CL in the following way.

Consider parent solution $C_p = \{C_1, \ldots, C_{K-1}, C_{p_{d1,d2}}\}$ and descendant solution $C_d = \{C_1, \ldots, C_{K-1}, C_{d1}, C_{d2}\}$ with parent cluster $C_{p_{d1,d2}}$ replaced by its descendants C_{d1} and C_{d2} ($C_{p_{d1,d2}} = C_{d1} \cup C_{d2}$). C_p and C_d represent a solution with K and $K+1$ clusters, respectively. $CL_p(\boldsymbol{X}, \Theta_K)$ for solution C_p can be written as:

$$CL_p(\boldsymbol{X}, \Theta_K) = \sum_{k}^{K-1} \sum_{x_i \in C_k} \log(p_k f(x_i|a_k)) + \sum_{x_i \in C_{p_{d1,d2}}} \log(p_{p_{d1,d2}} f(x_i|a_{p_{d1,d2}})) \tag{3}$$

$CL_d(\boldsymbol{X}, \Theta_{K+1})$ for solution C_d with $p_{d_1} + p_{d_2} = p_{p_{d1,d2}}$ can be written as:

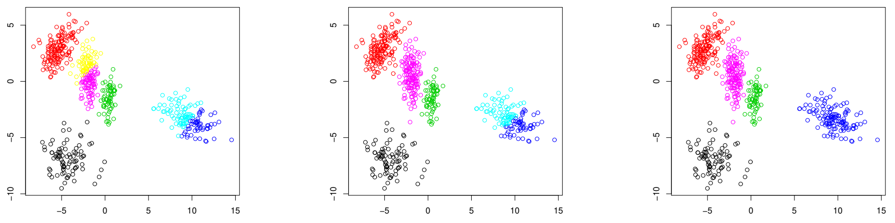
$$\begin{aligned} CL_d(\boldsymbol{X}, \Theta_{K+1}) = \sum_{k}^{K-1} \sum_{x_i \in C_k} \log(p_k f(x_i|a_k)) &+ \sum_{x_i \in C_{d_1}} \log(p_{d_1} f(x_i|a_{d_1})) \\ &+ \sum_{x_i \in C_{d_2}} \log(p_{d_2} f(x_i|a_{d_2})) \end{aligned} \tag{4}$$

The first part of Equations 3 and 4 is identical and can be ignored, if $CL_p(\boldsymbol{X}, \Theta_K)$ and $CL_d(\boldsymbol{X}, \Theta_{K+1})$ are compared. Therefore, comparing both solutions can be restricted to comparing the CL of the parent cluster ($C_{p_{d1,d2}}$) and the sum of the CLs of the descendants (C_{d1} and C_{d2}).

We claim that our method works well when each component in the initial GMM contains at most one mode from the data-generating distribution GMM_{data}. If this is not the case, it would not be possible to create a model with K components with one mode from GMM_{data} per component. However, the modes from GMM_{data} are presumably the densest parts of the data, leading to a separation of these in the leaf-level of the hierarchy. This property of HDBSCAN* hierarchies often holds in practice.

Figure 1 shows a dataset generated by a GMM with $K = 5$ and two merging iterations performed by our method, based on an initial GMM with $K = 7$. Figure 1a shows the clustering result according to the initial GMM after assigning all noise objects to the leaf clusters. We can observe that every leaf contains at most one mode from GMM_{data}. In Figure 1b the purple cluster resulted from merging two clusters since this resulted in a higher CL. Only one of them contained a mode. Similarly, Figure 1c shows the final clustering based on a GMM with $K = 5$. All final clusters contain exactly one mode. The subsequent potential merges all resulted in a solution with a lower, i.e. worse, CL.

The parameters of the candidate models (one for each parameterization) are refined again, running the CEM algorithm until convergence.



(a) Initial clustering. (b) Intermediate clustering. (c) Final result.

Fig. 1: Dataset generated from five Gaussians. Clusters contain between 50 and 100 objects. HDBSCAN* was run with $m_{pts} = 9$.

3. Model selection. After one candidate model for each possible value of m_{pts} and each parameterization is created in the previous step, the best has to be selected according to a model selection strategy. To do so, information criteria, such as the BIC and ICL, have been used in the literature to penalize larger models with more parameters.

Since it is not clear in which circumstances which criterion works best, we propose a model selection strategy similar to k-fold cross validation, apart from using the BIC or ICL. For this case, we still create candidate models as described in steps 1 and 2. However, in order to evaluate these models we split the dataset into m folds and refine them using the CEM algorithm m times on $m - 1$ folds (leaving a different fold out each time) and evaluate each model on the left-out fold using the loglikelihood. We can use the average of the resulting m loglikelihoods as criterion (denoted as VAL in the rest of this article) for model selection since they were computed using the left-out folds that were not used for fitting the model directly.

Therefore, BIC, ICL and VAL can be used as model selection criteria.

4 Evaluation

In this section, we present empirical results for our approach (denoted by HCEM) and two baselines. In order to measure the effectiveness of the approaches in recovering the data-generating distribution, we use synthetic datasets where this distribution is known as ground truth, and measure (1) the similarity of the GMMs discovered by the methods with the ground truth, and (2) the quality of the induced partition on the data. In addition, we use real datasets to show the superiority of our approach regarding the resulting partitions.

4.1 Baselines

Mclust [12] is based on EM and hierarchical initialization as described in Section 2.3 and uses the BIC for model selection. Since models up to n clusters (if there are n observations in the dataset) can be extracted from the agglomerative hierarchical clustering hierarchy, a set of possible $K \in \mathcal{K}$ has to be provided to restrict the number of components in the final set of models, from which the best according to BIC is selected. However, this requires domain knowledge and is difficult to set, if the approximate number of clusters is unknown, and computationally expensive, if $|\mathcal{K}|$ is large.

Mixmod [2] uses different traditional initialization methods, the BIC, ICL or NEC for model selection. We denote these different instances as Mixmod (BIC), Mixmod (ICL) and Mixmod (NEC), respectively.

4.2 Parameter settings

For our experiments, we set $\mathcal{K} = [2, K + 10]$ for Mclust and Mixmod, to make sure we include the correct number of clusters K in the dataset, while still keeping the range large enough to show that both baseline methods are able to select a model with K components from a larger collection. The three model parameterizations VVV, VVI and VII can be used by both methods.

In addition, Mixmod is initialized with the "smallEM" approach, running EM with random initialization 10 times for 5 iterations. It continues with the best model among these 10, according to the highest loglikelihood.

For HDBSCAN*, we chose $m_{pts} \in \{5, 6, 7, 8, 9, 10\}$, which are typical values. HCEM is run once for these values of m_{pts} and the result of the best model according to a model selection strategies is reported. As model selection strategies, we use BIC, ICL and VAL, denoting the resulting instances by HCEM (BIC), HCEM (ICL) and HCEM (VAL), respectively. Setting $\mathcal{K}$ is not required for HCEM since there is only a single model per hierarchy (one for each value of m_{pts}) created, limiting the set of models to be evaluated.

4.3 Synthetic datasets

We used the GMM data generator from [14] to create synthetic datasets with the following parameters:

– Number of Gaussians: 10, 40
– Dimensions: 2, 5, 10, 25, 50
– Objects per Gaussian: uniformly distributed in the range $[5 * D, 10 * D]$, where D is the dimensionality of the dataset.
– Mean: uniformly distributed in the range $[-10, 10]$
– Off-diagonal entries of covariance matrix: random number y in the range $[-1, 1]$, with a distribution following $y = x^2$ where x is a uniformly random deviate in $[0, 1]$ and the sign of y is determined randomly.
– Diagonal entries of covariance matrix: generated as the sum of all off-diagonal entries plus a random number y in the range $[0, 20 * \sqrt{D}]$ with a distribution following $y = x^2$, where x is a uniformly random deviate in $[0, 1]$, and the sign of y is determined randomly.

This generator [14] guarantees that generated Gaussians do not overlap. We created 20 datasets for each configuration (dimensionality and number of clusters) and averaged the result on these 20 datasets.

Model recovery results GMMs are versatile models that can be used to generate new data or as a classifier for unseen data. This leads to the goal of generating models that are ideally identical to the original data-generating distribution in their parameter settings. The Kullback-Leibler-Divergence (KLD) can be used as a measure of how much two GMMs deviate from each other (see [15]). However, the KLD is analytically not tractable, therefore [15] suggested to use Monte Carlo Sampling (MC) to compute an approximation:

$$D_{MC}\left(f \| g\right) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{f(x_i)}{g(x_i)} \tag{5}$$

Table 1 shows experiments on the output of Equation 5 using a sample of size $n = 100,000$ of the data generating distribution. f corresponds to the probability density of the original distribution from which the datasets were sampled, g corresponds to the probability density of the distribution fitted to the datasets by the three clustering algorithms. The lower the $D_{MC}\left(f \| g\right)$, the better, and a result of 0.0 means the distributions are identical.

The results were averaged over all 20 datasets per configuration, and we also report the overall average of all runs for all dimensions and number of clusters. Welch's t-test [21] for unequal variances was used with threshold 0.05 as significance test.

Mclust works well for most datasets, but is unstable in 50 dimensions, which lead to the second worst average out of all tested approaches. In order to merge the right clusters when building up the hierarchy for Mclust, accurate parameter

Dimensions	2		5		10		25		50		Avg
Alg/K	10	40	10	40	10	40	10	40	10	40	
Mclust(BIC)	**0.27**	0.50	0.31	0.35	0.37	**0.34**	**0.37**	**0.37**	2.68	0.66	0.62
Mixmod(BIC)	0.32	0.47	0.36	0.46	0.37	0.52	0.38	0.54	**0.40**	0.48	0.43
Mixmod(ICL)	0.31	0.45	0.32	0.43	0.37	0.54	0.38	0.60	**0.40**	0.60	0.44
Mixmod(NEC)	0.49	0.45	1.26	0.64	1.60	0.96	1.27	1.61	1.16	2.48	1.19
HCEM(BIC)	0.34	0.51	0.34	0.54	0.37	0.51	**0.37**	**0.37**	**0.40**	**0.40**	0.42
HCEM(ICL)	0.35	0.48	0.33	0.51	0.37	0.44	**0.37**	**0.37**	**0.40**	**0.40**	0.40
HCEM(VAL)	0.29	**0.40**	**0.28**	**0.33**	**0.34**	0.37	**0.37**	**0.37**	**0.40**	**0.40**	**0.35**

Table 1: Experiments on synthetic datasets, evaluated using KLD, averaged over 20 datasets per configuration. Lower is better; the best KLD is presented in **bold**. Results, that are statistically significant, are presented in green (better than all other approaches and model selection criteria) and blue (better than the other two approaches).

estimations are necessary, which is increasingly difficult with more dimensions for datasets of limited size.

While not creating the models with the best KLD in all but one configuration, Mixmod with ICL and BIC provides similar results to Mclust, except that it is able to return a lower KLD for 50-dimensional data. NEC performs worse than BIC and ICL in our setting with the only exception of 2 dimensions and 40 clusters. The average of all KLD values for Mixmod (BIC) and Mixmod (ICL) is higher than the worst HCEM approach, HCEM (BIC).

HCEM consistently returns the best GMM or close to the best, independently of the number of clusters in the dataset. There are only small differences between the models selected by BIC and ICL. HCEM (VAL) achieves overall the best average of 0.35 over all runs, which is statistically significant.

Clustering results We also evaluate the performance of all approaches using the Adjusted Rand Index (ARI) [16] to measure the quality of a clustering result. Table 2 shows experiments using Mclust, Mixmod and HCEM. The average ARI for all algorithms and data configurations (20 datasets each) is reported as well as the overall average all dimensions and number of clusters. Values in **bold** are the best results, i.e. highest ARI averaged over all 20 datasets per configuration.

The results lead to similar conclusions about the relative performance of the algorithms as in the previous experiments. All HCEM approaches achieve a better average ARI than Mclust and Mixmod for all model selection criteria. HCEM (VAL) achieves, again, the best average value and in 7 out of 10 configurations and the best individual average ARI. HCEM (BIC), HCEM (ICL) and HCEM (VAL) return, on average, better partitions than Mixmod and Mclust, and these differences are statistically significant.

Dimensions	2		5		10		25		50		Avg
Alg/K	10	40	10	40	10	40	10	40	10	40	
Mclust(BIC)	**0.92**	0.54	**1.00**	**0.98**	**1.00**	**1.00**	**1.00**	**1.00**	0.81	0.98	0.92
Mixmod(BIC)	0.89	0.54	0.97	0.93	0.98	0.95	0.99	0.95	**1.00**	0.96	0.92
Mixmod(ICL)	0.89	0.60	0.98	0.93	0.99	0.96	0.99	0.95	**1.00**	0.96	0.92
Mixmod(NEC)	0.76	0.66	0.64	0.78	0.63	0.76	0.85	0.72	0.91	0.63	0.73
HCEM(BIC)	**0.92**	0.68	0.99	0.94	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	0.95
HCEM(ICL)	**0.92**	0.71	0.99	0.96	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.96**
HCEM(VAL)	**0.92**	**0.74**	0.99	0.97	**1.00**	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**0.96**

Table 2: Experiments on synthetic datasets, evaluated using ARI, averaged over 20 datasets per configuration. Higher is better and best ARI is presented in **bold**. Results, that are statistically significant, are presented in green (better than all other approaches and model selection criteria) and blue (better than the other two approaches).

4.4 Real datasets

Here we present clustering results for 5 different real datasets from the UCI repository[4] with data statistics summarized in Table 3. The Digits dataset was downloaded from Scikit-learn [18]. Ecoli, Iris, Diabetes and Wine datasets were previously used for the evaluation of clustering algorithms, e.g. in [12], [5], [20] and [13]. Since the data-generating distribution is unknown for real data, we are using the ARI to judge the resulting partitions.

	Ecoli	Iris	Diabetes	Wine	Digits
Dimensions	7	4	3	13	64
Objects	336	150	145	178	1797

Table 3: Real datasets used in our experiments from the UCI repository.

We set $\mathcal{K} = [2, 100]$ for Mclust and Mixmod, which includes the correct number of clusters for each dataset. All other parameters are as described in Section 4.2. Table 4 shows experiments on all 5 datasets and compares the ARI of the resulting partitions using all available model parameterizations.

The results are consistent with the ARI experiments on synthetic data. For the Ecoli, Iris, Wine and Digits datasets, all HCEM approaches achieve a better clustering result than Mclust and all Mixmod approaches by a large margin. Mclust (BIC), Mixmod (BIC) and Mixmod (ICL) return the best result only on the Diabetes dataset, which is the dataset with the lowest dimensionality (only 3 dimensions). HCEM (BIC) returns the best partitions on three out of five cases for real data and is close to the best result on Diabetes and Iris.

[4] https://archive.ics.uci.edu/ml/datasets.php

Alg/Dataset	Ecoli	Iris	Diabetes	Wine	Digits
Mclust (BIC)	0.39	0.57	**0.66**	0.71	0.29
Mixmod (BIC)	0.34	0.31	**0.66**	0.68	0.30
Mixmod (ICL)	0.28	0.33	**0.66**	0.68	0.29
Mixmod (NEC)	0.07	0.57	0.44	0.08	0.27
HCEM (BIC)	**0.68**	0.90	0.65	**0.85**	**0.71**
HCEM (ICL)	0.66	**0.92**	0.53	0.83	0.62
HCEM (VAL)	0.66	**0.92**	0.53	0.83	0.62

Table 4: Experiments on real datasets, evaluated using ARI. Higher is better and best ARI is presented in **bold**.

5 Conclusion

We proposed a clustering framework for model-based clustering that is better in accurately recovering the data generating distribution and partitioning the data for synthetic as well as real datasets, compared to the baselines Mixmod and Mclust. Furthermore, the number of clusters does not have to be known a priori and no assumptions about the set $\mathcal{K}$ has to be made. We also showed that our approach is robust to different parameter settings (m_{pts} from HDBSCAN* as well as the model selection method). Therefore, HCEM seems to be a good choice for initializing the EM algorithm for clustering.

Currently 3 models are supported since only those make sure that models with different numbers of components can be extracted from the HDBSCAN* hierarchy efficiently without recomputing all model parameters after a decision about whether a parent or its descendant results in a better model. In [11] ways for efficiently extracting GMMs with a different number of components are described using more models than the three currently used ones. We leave implementing those for future work.

References

1. Biernacki, C., Celeux, G., Govaert, G.: Assessing a mixture model for clustering with the integrated completed likelihood. IEEE transactions on pattern analysis and machine intelligence **22**(7), 719–725 (2000)
2. Biernacki, C., Celeux, G., Govaert, G., Langrognet, F.: Model-based cluster and discriminant analysis with the mixmod software. Computational Statistics & Data Analysis **51**(2), 587–600 (2006)
3. Biernacki, C., Govaert, G.: Using the classification likelihood to choose the number of clusters. Computing Science and Statistics pp. 451–457 (1997)
4. 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 (2013)
5. Campello, R.J., Moulavi, D., Zimek, A., Sander, J.: Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM Transactions on Knowledge Discovery from Data (TKDD) **10**(1), 1–51 (2015)

6. Celeux, G., Govaert, G.: A classification em algorithm for clustering and two stochastic versions. Computational statistics & Data analysis **14**(3), 315–332 (1992)
7. Celeux, G., Govaert, G.: Gaussian parsimonious clustering models. Pattern recognition **28**(5), 781–793 (1995)
8. Celeux, G., Soromenho, G.: An entropy criterion for assessing the number of clusters in a mixture model. Journal of classification **13**(2), 195–212 (1996)
9. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the royal statistical society. Series B (methodological) pp. 1–38 (1977)
10. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd. vol. 96, pp. 226–231 (1996)
11. Fraley, C.: Algorithms for model-based gaussian hierarchical clustering. SIAM Journal on Scientific Computing **20**(1), 270–281 (1998)
12. Fraley, C., Raftery, A.E.: How many clusters? which clustering method? answers via model-based cluster analysis. The computer journal **41**(8), 578–588 (1998)
13. Gelbard, R., Goldman, O., Spiegler, I.: Investigating diversity of clustering methods: An empirical comparison. Data & Knowledge Engineering **63**(1), 155–166 (2007)
14. Handl, J., Knowles, J.: Cluster generators for large high-dimensional data sets with large numbers of clusters. Dimension **2**, 20 (2005)
15. Hershey, J.R., Olsen, P.A.: Approximating the kullback leibler divergence between gaussian mixture models. In: Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on. vol. 4, pp. IV–317. IEEE (2007)
16. Hubert, L., Arabie, P.: Comparing partitions. Journal of classification **2**(1), 193–218 (1985)
17. Neto, A.C.A., Sander, J., Campello, R.J., Nascimento, M.A.: Efficient computation of multiple density-based clustering hierarchies. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 991–996. IEEE (2017)
18. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
19. Schwarz, G., et al.: Estimating the dimension of a model. The annals of statistics **6**(2), 461–464 (1978)
20. Timm, H., Borgelt, C., Döring, C., Kruse, R.: An extension to possibilistic fuzzy cluster analysis. Fuzzy Sets and systems **147**(1), 3–16 (2004)
21. Welch, B.L.: The generalization ofstudent's' problem when several different population variances are involved. Biometrika **34**(1/2), 28–35 (1947)