Noname manuscript No.
(will be inserted by the editor)

Model-based Clustering using Automatic Differentiation: Confronting Misspecification and High-Dimensional Data

Siva Rajesh Kasa* · **Vaibhav Rajan**

Received: date / Accepted: date

Abstract We study two practically important cases of model based clustering using Gaussian Mixture Models: (1) when there is misspecification and (2) on high dimensional data, in the light of recent advances in Gradient Descent (GD) based optimization using Automatic Differentiation (AD). Our simulation studies show that EM has better clustering performance, measured by Adjusted Rand Index, compared to GD in cases of misspecification, whereas on high dimensional data GD outperforms EM. We observe that both with EM and GD there are many solutions with high likelihood but poor cluster interpretation. To address this problem we design a new penalty term for the likelihood based on the Kullback Leibler divergence between pairs of fitted components. Closed form expressions for the gradients of this penalized likelihood are difficult to derive but AD can be done effortlessly, illustrating the advantage of AD-based optimization. Extensions of this penalty for high dimensional data and for model selection are discussed. Numerical experiments on synthetic and real datasets demonstrate the efficacy of clustering using the proposed penalized likelihood approach.

Keywords Clustering · Gaussian Mixture Model · High-Dimensional Data · Misspecification · Automatic Differentiation

1 Introduction

Model-based clustering is a well-established paradigm for clustering multivariate data (Banfield and Raftery, 1993; Melnykov et al., 2010). In this approach, data is assumed to be generated by a finite mixture model where each component represents a cluster. For continuous-valued variables, it is common to model each component density by a multivariate Gaussian distribution. The rigorous probabilistic framework of mixture models facilitates model building and assessment. Gaussian Mixture Models (GMM), in particular, are widely used in a variety of applications (McLachlan and Peel, 2000). Expectation Maximization (EM) and its variants are by far the most popular methods to obtain Maximum Likelihood Estimates (MLE) of the parameters of a GMM (Dempster et al., 1977; McLachlan and Krishnan, 2007).

The one-to-one correspondence between fitted components and clusters, that makes model-based clustering intuitive and appealing, assumes that the underlying model is correctly specified and each data cluster can be viewed as a sample from a mixture component. In real data, the true distribution is rarely known and further, data may be contaminated by outliers from a distribution different from the assumed model. In such misspecified settings, MLE may be unrealistic and may fail to recover the underlying cluster structure (Farcomeni and Greco, 2016). Another related problem is that of spurious local maximizers. It is well known that the likelihood of GMM with unrestricted covariance matrices is unbounded (Day, 1969). As a result, one finds solutions from EM with fitted 'degenerate' component(s) having very small variance corresponding to a cluster containing very few closely located data points; in the case of multivariate data, there are components with very small generalized variance, lying in a lower-dimensional subspace and close to the boundary of the parameter space (Peel and McLachlan, 2000). They may have equal or higher likelihood compared to other solutions but without adequate interpretability in terms of cluster structure. Various constraints on the parameter space have been proposed to obtain non-spurious solutions through EM (Ingrassia, 2004; Ingrassia and Rocci,

*Corresponding Author

Department of Information Systems and Analytics
School of Computing, National University of Singapore
E-mail: kasa@u.nus.edu, vaibhav.rajan@nus.edu.sg

2007; Chen and Tan, 2009; Ingrassia and Rocci, 2011). Approaches to fit contaminated mixtures include trimming and use of constraints (Cuesta-Albertos et al., 1997; Ruwet et al., 2013; García-Escudero et al., 2014) and parsimonious models (Punzo and McNicholas, 2016). Recent theoretical results on misspecification in EM-based inference can be found in (Dwivedi et al., 2018).

Model-based clustering is also challenging for high dimensional data, particularly when the number of observations is relatively low. When the number of observations is small compared to the total number of free parameters to be estimated, the estimated covariance matrices are ill-conditioned and leads to unstable clustering (Bouveyron and Brunet-Saumard, 2014). When the number of observations is less than the data dimensionality, EM fails because the E-step requires the inversion of a matrix, that is typically not full-rank in such cases, and there are singularities in the parameter manifolds (Amari et al., 2006). Computational difficulties also arise due to quadratic increase in covariance matrix parameters with increasing dimensions. Common solutions include combining clustering with local dimensionality reduction, e.g., in the Mixture of Factor Analyzers (MFA) (Ghahramani and Hilton, 1997) and its extensions such as Parsimonious GMMs (McNicholas and Murphy, 2008) that impose constraints on the covariance structures. MLE for MFA and related models are also prone to spurious solutions and singularities (Greselin and Ingrassia, 2015; García-Escudero et al., 2016). Further, constraints on the covariance may lead to incorrect orientations or shapes of the resulting clusters (Zhou et al., 2009). A recent method by Cai et al. (2019), uses a modified E-step that obviates the need to compute the inverse of the covariance matrix but it assumes equi-covariance in all clusters and requires solving an expensive quadratic optimization problem. Another class of EM-based approaches are variable selection methods (Fop et al., 2018). Many of these approaches either fail or do not yield satisfactory clusters in high-dimensional low sample size regimes (Kasa et al., 2019). Further, in high dimensions likelihood-based criteria for model selection also fail (Giraud, 2014).

Gradient-based methods, as alternatives to EM, for MLE of GMMs have been studied previously (Redner and Walker, 1984; Xu and Jordan, 1996; Alexandrovich, 2014). Hybrid methods have also been proposed (Atkinson, 1992; Jamshidian and Jennrich, 1997). In general, methods based on conjugate gradients, quasi-Newton or Newton have been found to be inferior to EM (Xu and Jordan, 1996). These studies do not investigate cases of misspecification or high-dimensional settings. Our present work leverages Automatic Differentiation (AD) tools, that obviate the need to derive closed-form expressions of gradients and thereby facilitate Gradient Descent (GD) based inference[1] in complex models such as MFA. The computational efficiency and numerical accuracy offered by AD has been successfully leveraged in training over parameterized deep neural networks. Few previous works have investigated AD for optimization in statistical models – some examples include non-linear random effects models (Fournier et al., 2012; Skaug and Fournier, 2006; Skaug, 2002) and Bayesian models (Kucukelbir et al., 2017). AD-based Gradient Descent (AD-GD) for mixture models has been discussed previously in Maclaurin et al. (2015). However, to our knowledge, there is no previous work comparing the clustering performance of AD-GD with EM in the context of misspecification and high dimensional data, which is the focus of this paper.

Our extensive empirical analysis on simulated data shows that, in general, EM has better clustering performance (in terms of Adjusted Rand Index (ARI)) in cases of misspecification while AD-GD outperforms EM on high-dimensional data. However, we observe that, in misspecified cases, both EM and AD-GD can lead to poor clusterings for many different initializations. Similar to spurious solutions, these clusterings have high likelihood and low ARI, but they differ in other characteristics. For instance, we find that these solutions have components with large variance and occur frequently with many different initializations. With high dimensional data, we find that the use of AD-GD leads to components in the fitted GMM with low variance and high mixture weight (more details are in §3). To distinguish them from solutions previously characterized as spurious solutions, we call them *inferior* clusterings. To address the problem of inferior clusterings, we propose a new penalty term based on the Kullback Leibler divergence between pairs of fitted component Gaussians. Closed form expressions of the gradients for this penalty term are difficult to derive, but AD-based optimization can be done effortlessly. Extensions of this penalty for high-dimensional data and for model selection are discussed. Experiments on synthetic and real datasets demonstrate the efficacy of this penalized likelihood for cases of misspecification and high-dimensional data.

This paper is organized as follows. We begin with a brief overview of Mixture Models and Gradient Descent based inference using Automatic Differentiation (§2). We present our empirical study on clustering performance of EM and AD-GD on misspecified cases and high-dimensional data (§3). Our proposed penalty term is then introduced (§4) and the following section (§5) describes our clustering algorithms using this penalized likelihood. Next, we present the results

[1] Most AD solvers use minimization as the canonical problem and ML estimation is done by Gradient Descent on the negative loglikelihood. Throughout the paper, we use the term *Gradient Descent* for maximization problems as well, instead of Gradient Ascent, with the assumption that the sign of the objective function is changed during optimization.

of our simulation studies (§6). We discuss our experiments on real datasets (§7) and then conclude (§8).

2 Background

2.1 Mixture Models

Let $f(\boldsymbol{x};\boldsymbol{\theta})$ be the density of a K-component mixture model. Let f_k denote the kth component density with parameters $\boldsymbol{\theta}_k$ and weight π_k. The density of the mixture model is given by

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k f_k\left(\boldsymbol{x};\boldsymbol{\theta}_k\right),$$

where $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$ for $k = 1,\ldots,K$ and $\boldsymbol{\theta}$ denotes the complete set of parameters of the model. In a GMM, each individual component f_k is modeled using a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are its mean and covariance respectively. Appendix A has a list of symbols and notation used in the paper for reference.

Given n independent and identically distributed (iid) instances of p-dimensional data, $[x_{ij}]_{n\times p}$ where index i is used for observation, and j is used for dimension. Maximum Likelihood Estimation (MLE) aims to find parameter estimates $\hat{\boldsymbol{\theta}}$ from the overall parameter space $\boldsymbol{\Theta}$ of $f(\boldsymbol{\theta})$ such that probability of observing the data samples $\mathbf{x}_1,\ldots,\mathbf{x}_n$ from this family is maximized, i.e., $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\theta})$, where, $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n}\sum_i \log f(\mathbf{x}_i;\boldsymbol{\theta})$ is the loglikelihood.

Following White (1982), if the observed data are n iid samples from a probability distribution $P(\eta^*)$ (where η^* denotes the *true* set of parameters) and the fitted model has the same functional form $P(.)$, then the model is said to be correctly specified. Otherwise, the model is said to be *misspecified*. An example of misspecification is when the data is sampled from a multivariate t-distribution, and clustered using a GMM. It is easy to see that many real-world applications of model-based clustering are prone to misspecification.

2.2 Mixture of Factor Analyzers (MFA)

Without any restrictions, the GMM is a highly parameterized model where $K-1$ mixture weights, Kp elements of the mean vectors and $\frac{1}{2}Kp(p+1)$ covariance matrix elements have to be estimated which can be difficult in high dimensions. MFA provides a parsimonious characterization through local dimensionality reduction using q-dimensional latent factors $\mathbf{F}_i$ that model correlations in the data, where $q < p$ (Ghahramani and Hilton, 1997). Each observation $\mathbf{x}_i$ (for $i = 1,\ldots,n$) is assumed to be generated as follows:

$$\mathbf{x}_i = \boldsymbol{\mu}_k + \boldsymbol{\Lambda}_k\mathbf{F}_{ik} + \mathbf{e}_{ik} \quad \text{with probability } \pi_k \tag{1}$$

where $\boldsymbol{\Lambda}_k$ is a $p\times q$ matrix of *factor loadings* and both the *factors* $\mathbf{F}_{ik}$ and *errors* $\mathbf{e}_{ik}$ follow Gaussian distributions that are independent of each other:

$$\mathbf{F}_{1k},\ldots,\mathbf{F}_{nk} \sim \mathcal{N}(\mathbf{0},\mathbb{I}_q); \qquad \mathbf{e}_{ik} \sim \mathcal{N}(\mathbf{0},\boldsymbol{\Psi}_k)$$

where $\mathbb{I}_q$ is the q-dimensional Identity matrix and $\boldsymbol{\Psi}_k$ is a $p\times p$ diagonal matrix ($k = 1,\ldots,K$). Thus, the component covariance matrices have the form $\boldsymbol{\Sigma}_k = \boldsymbol{\Lambda}_k\boldsymbol{\Lambda}_k^{'} + \boldsymbol{\Psi}_k$. Since $\frac{1}{2}q(q-1)$ constraints are required to uniquely define $\boldsymbol{\Lambda}_k$, the number of free parameters to be estimated is $K(pq - \frac{1}{2}q(q-1) + p)$. See, e.g., Greselin and Ingrassia (2015) for details.

2.3 Model Selection

Criterion-based model selection methods are designed to favor parsimony in modeling, by penalizing overfitting of the model. Common likelihood-based criteria include Akaike Information Criterion (AIC):

$$\text{AIC} := 2p_e - 2\mathcal{L}(\hat{\boldsymbol{\theta}}),$$

and Bayesian Information Criterion (BIC):

$$\text{BIC} := p_e\ln(n) - 2\mathcal{L}(\hat{\boldsymbol{\theta}}),$$

where p_e is the number of parameters to be estimated in the model.

It has been pointed out that MLE based criteria such as AIC and BIC are based on asymptotic normality and hence, cannot be used for model selection in high-dimensions where $n << p$ (Amari et al., 2006; Akaho and Kappen, 2000; Giraud, 2014). Moreover, in high dimensions, it has been observed that BIC tends to underestimate the number of components whereas AIC overestimates the number of components (Melnykov et al., 2010).

2.4 Spurious Solutions

Spurious solutions are local maximizers of the likelihood function but lack real-life interpretability and hence do not provide a good clustering of the data. It is a consequence of the unboundedness of the GMM likelihood function for unrestricted component covariance matrices. As discussed in McLachlan and Peel (2000), spurious solutions may be obtained when:

- a fitted component has very small non-zero variance for univariate data or generalized covariance, i.e., the determinant of the covariance matrix, for multivariate data. Such a component corresponds to a cluster containing very few data points close together or, for multivariate data, in a lower dimensional subspace.

- the model fits a small localized random pattern in the data instead of an underlying group structure. Such solutions have very few points in one cluster with little variation, compared to other clusters, in the cluster's axes, or for multivariate data, small eigenvalues for the component covariance matrix.
- where the likelihood increases by fitting the covariance matrix of a component on just one or a few datapoints distant from the other samples.

In some cases, e.g., when a component is fitted over very few datapoints, spurious solutions lead to singularities in the component covariance matrices that can be detected during EM inference. There are other cases as well, when the parameters lie close to the boundary of the parameter space, when the component covariance matrices are not singular but may be close to singular for some components.

McLachlan and Peel (2000) observed that convergence to spurious solutions happen rarely; they are dependent on initialization – often occurring only for some initializations; and convergence to non-spurious global maxima becomes difficult with increasing dimensionality.

Both noise and outliers can lead to spurious solutions. Contaminating outliers from a population different from the assumed model are called mild outliers (Ritter, 2014)). If additional procedures, such as trimming (Cuesta-Albertos et al., 1997; García-Escudero et al., 2010) are not performed, then fitting a GMM leads to misspecification.

2.5 Gradient Descent and Automatic Differentiation

In traditional GD based inference, gradients are required in closed form which becomes laborious or intractable to derive as the model complexity increases, e.g., for MFA. To automate the computation of derivatives three classes of techniques have been developed: (a) Finite Differentiation (FD) (b) Symbolic Differentiation (SD) and (c) Automatic or Algorithmic Differentiation (AD). Although easy to code, FD is slow at high dimensions and susceptible to floating point errors. SD provides exact symbolic expressions of derivatives but has high computational complexity, in both time and memory and cannot be used when functions are defined using programmatic constructs such as conditions and loops. The complexity and errors in both FD and SD increase for the computation of higher derivatives and partial derivatives of vector-valued functions. AD overcomes all these limitations of FD and SD. It provides efficient and accurate numerical evaluation of derivatives without requiring closed form expressions.

The numerical computation of a function can be decomposed into a finite set of elementary operations. These operations most commonly include arithmetic operations and transcendental function evaluations. The key idea of AD is to numerically compute the derivative of a function by combining the derivatives of the elementary operations through the systematic application of the chain rule of differential calculus. The efficiency of the computation is improved by storing evaluated values of intermediate sub-expressions that may be re-used. Backpropagation, used to train neural networks, is a specific form of AD which is more widely applicable. We refer the reader to recent surveys (Baydin et al., 2018; Margossian, 2019) for more details. Appendix B discusses some examples.

Efficient implementations of AD are available in several programming languages and frameworks, e.g. Python (Maclaurin et al., 2015), R (Pav, 2016), Pytorch (Paszke et al., 2017) and Stan (Carpenter et al., 2015). Many first and second order gradient-based optimization algorithms are implemented in these libraries. In this paper, we use Adam, a first order method that computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients (Kingma and Ba, 2015).

3 Automatic Differentiation for Mixture Model based Clustering

In this section we first summarize how Automatic Differentiation based Gradient Descent (AD-GD) can be used for GMM based clustering. Second, we discuss our numerical studies that compare the performance of EM and AD-GD in cases of misspecification and high-dimensions. Third, we explain the problems arising in GMM based clustering that are unresolved by both AD-GD and EM, which motivate the development of our new algorithm in the following section.

3.1 Automatic Differentiation based Gradient Descent (AD-GD)

To obtain MLE of GMMs EM elegantly solves 3 problems: (1) Intractability of evaluating the closed-forms of the derivatives, (2) Ensuring positive definiteness (PD) of the covariance estimates $\hat{\mathbf{\Sigma}}_k$, and (3) Ensuring the constraint on the component weights ($\sum_k \hat{\pi}_k = 1$).

In AD-GD, *Problem 1* is solved inherently because we do not need to express the gradients in closed form by virtue of using AD. Further, the second-order Hessian matrix may also be evaluated using AD, enabling us to use methods with faster convergence. To tackle *Problem 2*, instead of gradients with respect to $\mathbf{\Sigma}_k$, we compute the gradients with respect to $\mathbf{U}_k$, where $\mathbf{\Sigma}_k = \mathbf{U}_k\mathbf{U}_k^T$. We first initialize $\mathbf{U}_k$ as identity matrices. Thereafter, we keep adding the gradients to the previous estimates of $\hat{\mathbf{U}}_k^{t+1}$, i.e.

$$\hat{\mathbf{U}}_k^{t+1} := \hat{\mathbf{U}}_k^t + \epsilon\frac{\partial\mathcal{L}}{\partial\mathbf{U}_k};\ \ \hat{\mathbf{\Sigma}}_k^{t+1} := \hat{\mathbf{U}}_k^{t+1}\hat{\mathbf{U}}_k^{t+1^T} \tag{2}$$

Algorithm 1 AD-GD

Input: Data: $n \times p$ dimensional matrix, required number of clusters K, learning rate ϵ and convergence tolerance γ.
Initialize at iteration $t = 0$**:** $\hat{\boldsymbol{\mu}}_k^0$, $\hat{\alpha}_k$'s using K-Means or random initialization; $\hat{\mathbf{U}}_k^0$ (for GMM) or $\hat{\boldsymbol{\Lambda}}_k^0$, $\hat{\boldsymbol{\Psi}}_k^0$ (for MFA) as identity matrices.
Objective: $\text{argmax}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$
REPEAT: At every iteration $t+1$:
Mixture weight and mean updates

$$\hat{\alpha}_k^{t+1} := \hat{\alpha}_k^t + \epsilon \frac{\partial \mathcal{L}}{\partial \alpha_k}; \; \hat{\pi}_k^{t+1} := \frac{e^{\hat{\alpha}_k^{t+1}}}{\sum_{k'} e^{\hat{\alpha}_{k'}^{t+1}}}; \; \hat{\boldsymbol{\mu}}_k^{t+1} := \hat{\boldsymbol{\mu}}_k^{t+1} + \epsilon \frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k};$$

Covariance updates
– For GMM:

$$\hat{\mathbf{U}}_k^{t+1} := \hat{\mathbf{U}}_k^t + \epsilon \frac{\partial \mathcal{L}}{\partial \mathbf{U}_k}; \; \hat{\boldsymbol{\Sigma}}_k^{t+1} := \hat{\mathbf{U}}_k^{t+1} \hat{\mathbf{U}}_k^{t+1^T}$$

– For MFA:

$$\hat{\boldsymbol{\Lambda}}_k^{t+1} := \hat{\boldsymbol{\Lambda}}_k^t + \epsilon \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Lambda}_k}; \hat{\boldsymbol{\Psi}}_k^{t+1} := \hat{\boldsymbol{\Psi}}_k^t + \epsilon \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Psi}_k}; \hat{\boldsymbol{\Sigma}}_k^{t+1} := \hat{\boldsymbol{\Lambda}}_k^{t+1} \hat{\boldsymbol{\Lambda}}_k^{t+1^T} + \hat{\boldsymbol{\Psi}}_k^{t+1}$$

UNTIL: convergence criterion $|\mathcal{L}^{t+1} - \mathcal{L}^t| < \gamma$ is met

where ϵ is the learning rate and superscripts $t, t+1$ denote iterations in GD. If the gradients are evaluated with respect to $\boldsymbol{\Sigma}_k$ directly, there is no guarantee that updated $\hat{\boldsymbol{\Sigma}}_k^{t+1} = \hat{\boldsymbol{\Sigma}}_k^t + \epsilon \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k}$ will still remain PD. However, if the gradients are evaluated with respect to $\mathbf{U}_k$, by construction $\hat{\boldsymbol{\Sigma}}_k^{t+1}$ will always remains PD. Cholesky decomposition for reparameterizing $\boldsymbol{\Sigma}_k$ can also be used (Salakhutdinov et al., 2003). In the case of MFA models, the component covariance matrices are in factorized form, $\boldsymbol{\Sigma}_k = \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^T + \boldsymbol{\Psi}_k$, which ensures PD covariance matrix estimates. *Problem 3* is solved by using the log-sum-exp trick (Robert, 2014). We start with unbounded α_k as the log-proportions: $\log \pi_k = \alpha_k - \log(\sum_{k'=1}^K e^{\alpha_{k'}})$. We need not impose any constraints on α_k as the final computation of π_k automatically leads to normalization, because $\pi_k = \frac{e^{\alpha_k}}{\sum_{k'=1}^K e^{\alpha_{k'}}}$. Therefore, we reparametrize a constrained optimization problem into an unconstrained one (without using Lagrange multipliers) and we can update $\hat{\pi}_k$ as follows:

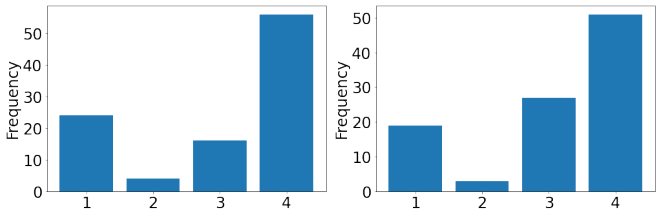
$$\hat{\alpha}_k^{t+1} := \hat{\alpha}_k^t + \epsilon \frac{\partial \mathcal{L}}{\partial \alpha_k}; \;\; \hat{\pi}_k^{t+1} := \frac{e^{\hat{\alpha}_k^{t+1}}}{\sum_{k'=1}^K e^{\hat{\alpha}_{k'}^{t+1}}} \tag{3}$$

Algorithm 1 shows all the steps for AD-GD inference of GMM and MFA.

3.2 Misspecification

We study the clustering solutions obtained by fitting misspecified GMM on Pinwheel data, also called warped GMM, with 3 components and 2 dimensions (Iwata et al., 2013). Pinwheel data is generated by sampling from Gaussian distributions, and then stretching and rotating the data. The centers are equidistant around the unit circle. The variance is controlled by two parameters r and t, the radial standard deviation and the tangential standard deviation respectively. The warping is controlled by a third parameter, s, the rate parameter. Thus, the extent of misspecification (i.e., the deviation of the data from the assumed Gaussian distributions in GMM-based clustering) can be controlled using these parameters. An example is shown in fig. 2. We generate 1800 Pinwheel datasets with different combinations of parameters. In addition, we also simulate 1800 3-component, 2-dimensional datasets from GMM with varying overlap of components (to analyze the case where there is no misspecification in fitting a GMM). Details of both Pinwheel and GMM datasets used are given in Appendix C.1. For each dataset we obtain two clustering solutions by using EM and AD-GD. We run both the algorithms using the same initialization and stopping criterion (convergence threshold $1e-5$). We use ARI (Hubert and Arabie, 1985) to evaluate performance where higher values indicate better clustering.

Our extensive experiments on these 3600 datasets, described in Appendix C.1, show that EM outperforms AD-GD in both cases – when there is misspecification and no missspecification. However, when there is misspecification, we find that in both EM ad AD-GD, there are many *inferior* solutions that have low ARI and unequal fitted covariances, often with one fitted component having relatively large covariance, resulting in a high degree of overlap between components. We illustrate this using Pinwheel data generated with parameters $r = 0.3$, $t = 0.05$ and $s = 0.4$. Both EM and AD-GD are run with 100 different initializations. Table 1 shows the summary statistics of the AD-GD solutions where we have grouped the solutions into 4 sets. Appendix C.4 has statistics of the EM solutions that are very similar. We categorize all the solutions with AIC higher than 788 into the fourth set. Fig. 1 shows the histogram of the counts in the 4 sets. We visualize the clustering for a single solution from sets 1–3 and 5 solutions from set 4 in fig. 2.



(a) EM (b) AD-GD

Fig. 1: Frequency over 100 random initializations

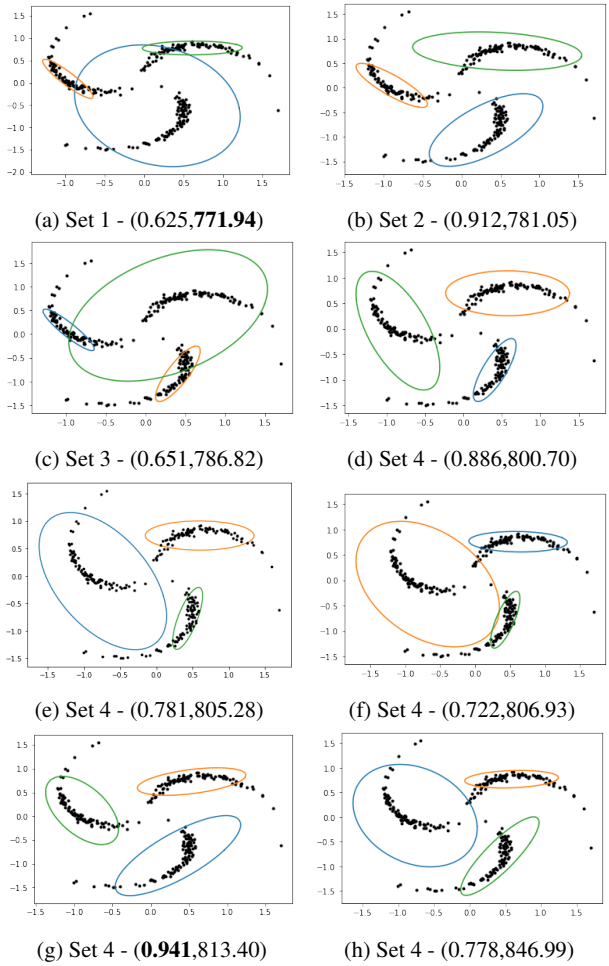
We observe that both EM and AD-GD obtain very similar solutions in terms of AIC and ARI for this dataset. The best average ARI is that of set 2 (row 2 of Table 1). Fig. 1 shows that these solutions are obtained in less than 5% of the cases. The overall best ARI, of 0.941, is from a solution in set 4 that has a high AIC value of 813.4 as shown in fig.

Table 1: Summary statistics of AD-GD clustering solutions over 100 random initializations on the Pinwheel dataset (shown in fig. 2), grouped into 4 sets based on AIC ranges. Mean and standard deviation of AIC, ARI, component weights and covariance determinants, over solutions in each set.

Set	AIC Range	AIC	ARI	π_1	π_2	π_3	$\lvert\mathbf{\Sigma}_1\rvert$	$\lvert\mathbf{\Sigma}_2\rvert$	$\lvert\mathbf{\Sigma}_3\rvert$
1	771-773	771.9	0.625	0.257	0.265	0.477	0.0002	0.0005	0.123
		(6e-8)	(2e-16)	(3e-6)	(3e-6)	(1e-6)	(1e-9)	(4e-9)	(4e-7)
2	781-782	781.1	0.912	0.306	0.341	0.352	7e-4	0.01	0.01
		(3e-6)	(0)	(1e-5)	(3e-5)	(1e-5)	(5e-9)	(5e-9)	(5e-7)
3	786-787	786.8	0.652	0.257	0.267	0.475	2e-4	5e-4	0.156
		(2e-7)	(0)	(5e-6)	(3e-6)	(3e-6)	3e-9	(3e-9)	(2e-6)
4	788-850	815.0	0.806	0.28	0.315	0.403	6e-4	3e-3	0.047
		(17.84)	(0.06)	(0.015)	(0.010)	(0.024)	(4e-4)	(1e-3)	(0.018)

2 (g). The best AIC, of 771.9, is obtained by a solution from set 1 which has considerably lower ARI of 0.625 as shown in fig. 2 (a). Thus, high likelihood does not correspond to a good clustering. We observe that there are many inferior solutions in sets 1, 3 and 4 having a fitted component with large variance and low ARI, also seen in the specific solutions shown in fig. 2.

We find that such inferior solutions with low ARI and low AIC in misspecified models occur frequently with many different initializations, and typically when there is a component with large variance. This is different from the characterization of spurious solutions that were found to occur rarely, only with certain initializations and due to a component with small variance (McLachlan and Peel, 2000).



(a) Set 1 - (0.625,**771.94**) (b) Set 2 - (0.912,781.05)

(c) Set 3 - (0.651,786.82) (d) Set 4 - (0.886,800.70)

(e) Set 4 - (0.781,805.28) (f) Set 4 - (0.722,806.93)

(g) Set 4 - (**0.941**,813.40) (h) Set 4 - (0.778,846.99)

Fig. 2: 8 clustering solutions obtained with AD-GD using different initializations on Pinwheel data (with parameters $r = 0.3, t = 0.05, s = 0.4$); Sets refer to groups defined in Table 1; in parentheses: (ARI, AIC), best values in bold.

3.3 High Dimensional Data

EM-based methods fail to run in high-dimensional low sample size regime. We observe that the clustering solutions from AD-GD often have a single component with relatively low variance and high mixture weight spread over all the datapoints. We call such components 'dominating' components. While the final dominating component depends on the initialization, one such component is present for almost all the initializations. Note that while the characterization of this clustering is also different from that of previously studied spurious solutions, the effect remains the same – that of an inferior clustering. Here the likelihood increases predominately by increasing the mixture component weight of the dominating component.

We illustrate the dominating components through the determinants of covariance matrices, using a simulated 200-dimensional 4-component dataset. Each component has covariance matrix $\mathbf{\Sigma}_k = 0.5 * \mathbb{I}$. The mean of the components are chosen as shown in table 2. For each component, we sample 15 datapoints.

Fig. 3 illustrates the dominating component phenomenon over multiple iterations: soon after initialization, the variance of one of the components (the dominating component) increases to capture all the near-by points and then, the variance of this component is reduced while at the same time the corresponding component weight is increased, to maximize the likelihood.

Table 2: Component (C) means

Features	C-1	C-2	C-3	C-4
1-20	1	0	0	0
20-40	0	1	0	0
40-60	0	0	1	0
60-80	0	0	0	1
80-200	0	0	0	0

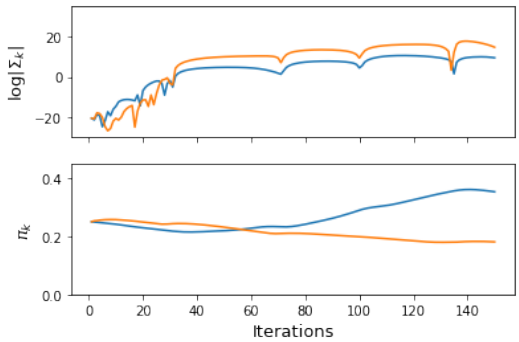


Fig. 3: In HD ($p > n$): (a) maximizing likelihood leads to reduction in variance (above) and corresponding increase in weight (below) resulting in a *dominating component* (in blue). GMM (AD): ARI = 0.037 Loglikelihood = -24416.

4 A Penalty Term based on KL Divergence

Our empirical analysis in the previous section reveals that clustering solutions from AD-GD tend to have inferior solutions in cases of misspecification and high dimensional data. Although their characteristics are different from previously studied spurious solutions, they are similar in terms of being poor clustering solutions that have high likelihood. A well-known technique to avoid spurious solutions is to constrain the parameter space (Hathaway, 1985)). Approaches include restrictions on scale and weight parameters, e.g., through bounds on the minimum eigenvalues of component covariance matrices (Ingrassia and Rocci, 2007, 2011) or constraining their determinants (Biernacki, 2004) during EM estimation. An approach based on penalized ML estimators was proposed by Chen and Tan (2009). They design a family of penalty terms that ensures consistency of the estimator. An example of a penalty term that belongs to their family is: $\sum_{k=1}^{K} -\frac{1}{n}\left(\text{tr}\left(\boldsymbol{\Sigma}_k^{-1}\right) + \log|\boldsymbol{\Sigma}_k|\right)$. We design a penalty term for the likelihood that is similar in spirit, but more general, to address the problem of inferior clustering solutions.

4.1 Motivating Examples

In cases of misspecification, inferior clusterings from both EM and AD-GD, are found to have fitted covariances that differ considerably in their orientations and sizes. We observe this in the solutions in fig. 2

As another motivating example, consider a case where misspecification arises due to contamination (Punzo and McNicholas, 2016). Consider a 2-dimensional GMM with 4 components, each with unit covariance. The means of the four components are set as given in table 3 with $\beta = 3$. From each component, we sample 50 datapoints. Now, we

Table 3: Component (C) Means

	C-1	C-2	C-3	C-4
μ_1	-β	0	0	β
μ_2	0	β	-β	0

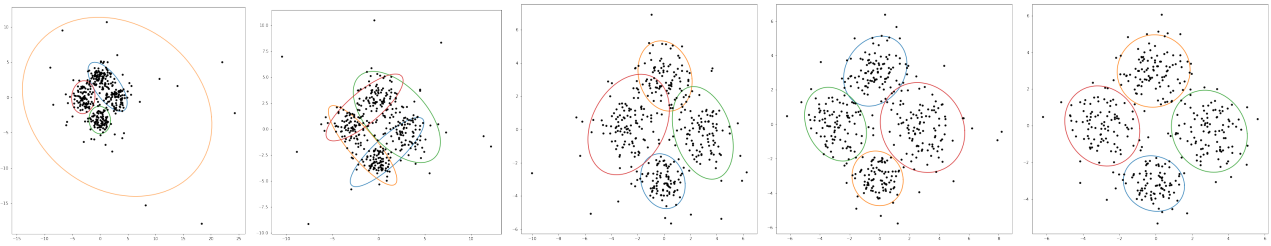
consider a 2-dimensional 4-component multivariate t distribution with same means and covariance structures. For the mixture of multivariate t distribution also, we sample 50 datapoints from each component. Each dataset has 400 samples. We thus have a contaminated Gaussian mixture. When the degree of freedom is high, the contamination is negligible. When the degree of freedom is low, the t-distributions components will have heavier tails, hence the contamination is higher. By varying the degrees of freedom we can control the contamination and hence the misspecification. We fit a 4-component GMM using EM (We obtain similar results using AD-GD as well). As seen from figure 4, the components tend to overlap with increasing misspecification (decreasing D.o.F). We observe that when the misspecification is high, some of the fitted components tend to have higher covariance compared to other components.

We repeat the same experiment of corrupting the GMM with a multivariate t-distribution but with increasing the cluster separation by setting $\beta = 5$. The results are shown in fig. 5. As seen, increasing cluster separation mitigates the affect of misspecification. A similar experiment on contamination with random noise is shown in Appendix C.2.

4.2 Combinatorial KL Divergence

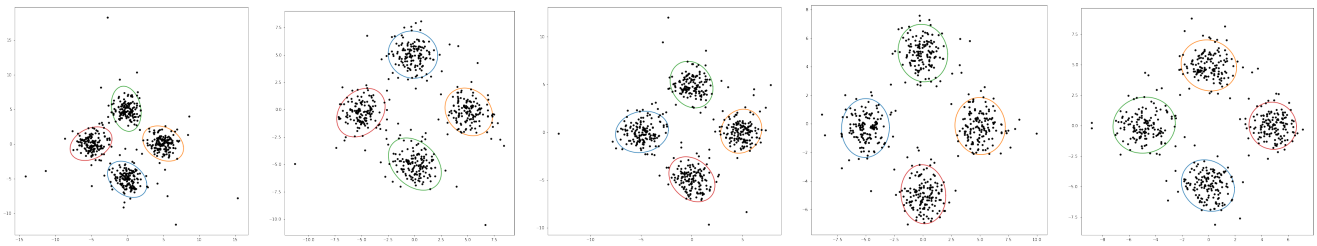
We propose a novel penalty term that penalizes differences in orientations and sizes of the fitted components as well as degenerate solutions. Our penalty term is based on the KL-divergence between component Gaussians. Let $\mathcal{N}_k$ denote the multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. The KL-divergence between $\mathcal{N}_1$ and $\mathcal{N}_2$ is given by:

$$
\begin{aligned}
&KL\left(\mathcal{N}_1, \mathcal{N}_2\right) \\
&\quad = KL\left(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\right) \\
&\quad = \frac{1}{2}\Bigg[\log\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} - p \\
&\qquad\qquad + \text{tr}\{\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\} + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\Bigg]
\end{aligned}
$$



(a) D.o.F = 2 (b) D.o.F = 3 (c) D.o.F = 4 (d) D.o.F = 5 (e) D.o.F = 10

Fig. 4: Lower the degree of freedom (D.o.F), higher the misspecification, greater the difference in orientation and size of the components fitted



(a) D.o.F = 2 (b) D.o.F = 3 (c) D.o.F = 4 (d) D.o.F = 5 (e) D.o.F = 10

Fig. 5: When the cluster separation is high, the misspecification does not seem to affect the fitted components much - only the orientation of the fitted components is affected

Table 4: Purpose served by each term in $KL\left(\mathcal{N}_1, \mathcal{N}_2\right)$

Penalty term	Purpose
$\mathrm{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2) - p$	This term penalizes the difference in orientations, i.e. if the directions of principal axes of the covariance matrices of the two components are vastly different. When $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are exactly similar, this penalty term goes to zero.
$\log \frac{\lvert\boldsymbol{\Sigma}_2\rvert}{\lvert\boldsymbol{\Sigma}_1\rvert}$	This term penalizes the difference in size of the covariance matrices of the two components. Note that this term can go to zero even if the orientations of two covariance matrices are different.
$(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_2^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$	This term penalizes the assignment of a single cluster to far-away outlier points

Each term in the expression above can provide useful constraints as summarized in Table 4.

KL-divergence is not symmetric about $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$. If there is an order of magnitude difference in the covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, it can be detected through the values of $KL\left(\mathcal{N}_1, \mathcal{N}_2\right)$ and $KL\left(\mathcal{N}_2, \mathcal{N}_1\right)$. The difference in their values is primarily contributed by the difference between the terms $(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_2^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$ and $(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$, and $\mathrm{tr}\{\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\}$ and $\mathrm{tr}\{\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_2\}$. The difference in these two KL divergence values provides signal about the overlap or asymmetry of the two Gaussians. This notion is generalized to a K-component GMM, through the combinatorial KL divergences, KLF (forward) and KLB (backward):

$$KLF := \sum_{1 \le k_1 < k_2 \le K} KL\left(\mathcal{N}_{k_1}, \mathcal{N}_{k_2}\right); \tag{4}$$

$$KLB := \sum_{1 \le k_2 < k_1 \le K} KL\left(\mathcal{N}_{k_1}, \mathcal{N}_{k_2}\right). \tag{5}$$

Well separated clusters typically have equal and lower values of KLF and KLB. We denote both the values by

$$KLDivs = \{KLF, KLB\}.$$

We note that these two sums (KLF + KLB) together give the sum of Jeffrey's divergence between all components. In the clustering outputs shown in fig. 2, we see that in solution (c) from set 3, where clustering is poor, KLF$=258$ and KLB$=494$ (the difference is high), while in solution (g) from set 4, which has better clustering, KLF$=127$ and KLB$=64$ (with relatively lower difference).

Our proposed penalty term is a weighted sum of the KLF and KLB terms:

$$-w_1 \times KLF - w_2 \times KLB \tag{6}$$

Notice that the weights $(-w_1, -w_2)$ are negative. If we add positive weights GD will further shrink the smaller clusters. Negative weights lead to separation as well as clusters of

similar volume. In high dimensions, the variance of the non-dominating components increase to reduce KLF and KLB (for an example, see section 5.5). To tackle this problem, we add another term to penalize very small covariance matrices: $-w_3 \times \sum_{k=1}^{K}(\det(\mathbf{\Sigma}_k) - \lambda_k)^2$.

4.3 Implementation

Optimization of likelihood with these penalty terms can be implemented effortlessly in the AD-GD framework where gradients in closed forms are not required. However, the use of such complex penalties is difficult within EM. We cannot obtain closed forms of the covariance estimates. Closed forms for the mean update can be derived (Appendix D) but is laborious. Further the update for the mean of a component depends on means for all other components, and hence cannot be parallelized in EM.

5 Sequential Initialization Algorithm

Our algorithm is similar in principle to that of Biernacki et al. (2003) who suggest the use of initializing EM with a few short runs of EM. In our algorithms, after the first run that can use EM or AD-GD without the penalty term, we run AD-GD with the penalty term. Note that for high-dimensional data, the first run requires AD-GD.

5.1 SIA

SIA consists of two steps. In the first step of SIA, we use the loglikelihood $\mathcal{L}$ as the objective and run EM or AD-GD (first or second order) to fit a GMM. Typically, the output at the end of first step will have unequal KL-divergences for misspecified models. We take these parameters at the end of the first step to initialize the algorithm in the second step. In the second step we modify the objective function to:

$$\mathcal{M} = \mathcal{L} - w_1 \times KLF - w_2 \times KLB \tag{7}$$

After the second optimization step the likelihood decreases but the KL-divergence values, KLF and KLB, come closer. Thus there is a trade off between likelihood and cluster separation. In high-dimensions, the objective function $\mathcal{M}$ is set to:

$$\mathcal{M} = \mathcal{L} - w_1 \times KLF - w_2 \times KLB - w_3 \times \sum_{k=1}^{K}(\det(\mathbf{\Sigma}_k) - \lambda_k)^2 \tag{8}$$

To distinguish between the low-dimensional and high-dimensional cases, we call the latter SIA-HD. The complete algorithm is presented in Algorithm 2.

Algorithm 2 SIA

Input: Data: $n \times p$ dimensional matrix, required number of clusters K, learning rate ϵ, convergence tolerance γ.
Initialize at iteration $t = 0$**:** $\hat{\boldsymbol{\mu}}_k^0$, $\hat{\alpha}_k$'s using K-Means or random initialization; $\hat{\mathbf{U}}_k^0$ (for GMM) or $\hat{\mathbf{\Lambda}}_k^0$, $\hat{\mathbf{\Psi}}_k^0$ (for MFA) as identity matrices.
Step I: Run Algorithm 1 (AD-GD)
Step II:
For Low Dimensional Data: Set
$\mathcal{M} = \mathcal{L} - w_1 \times KLF - w_2 \times KLB$
For High-Dimensional Data: Use the determinant of $\mathbf{\Sigma}_k$'s to identify dominating components and set penalty terms λ_k. Set

$$\mathcal{M} = \mathcal{L} - w_1 \times KLF - w_2 \times KLB - w_3 \times \sum_{k=1}^{K}(\det(\mathbf{\Sigma}_k) - \lambda_k)^2$$

Initialize: Use Output of Step I
REPEAT: At every iteration $t+1$:

$$\hat{\alpha}_k^{t+1} := \hat{\alpha}_k^t + \epsilon\frac{\partial \mathcal{M}}{\partial \alpha_k};\ \hat{\pi}_k^{t+1} := \frac{e^{\hat{\alpha}_k^{t+1}}}{\sum_{k'} e^{\hat{\alpha}_{k'}^{t+1}}};\ \hat{\boldsymbol{\mu}}_k^{t+1} := \hat{\boldsymbol{\mu}}_k^{t+1} + \epsilon\frac{\partial \mathcal{M}}{\partial \boldsymbol{\mu}_k};$$

Covariance updates for GMM:

$$\hat{\mathbf{U}}_k^{t+1} := \hat{\mathbf{U}}_k^t + \epsilon\frac{\partial \mathcal{M}}{\partial \mathbf{U}_k};\ \hat{\mathbf{\Sigma}}_k^{t+1} := \hat{\mathbf{U}}_k^{t+1}\hat{\mathbf{U}}_k^{t+1^T}$$

Covariance updates for MFA:

$$\hat{\mathbf{\Lambda}}_k^{t+1} := \hat{\mathbf{\Lambda}}_k^t + \epsilon\frac{\partial \mathcal{M}}{\partial \mathbf{\Lambda}_k};\hat{\mathbf{\Psi}}_k^{t+1} := \hat{\mathbf{\Psi}}_k^t + \epsilon\frac{\partial \mathcal{M}}{\partial \mathbf{\Psi}_k};\hat{\mathbf{\Sigma}}_k^{t+1} := \hat{\mathbf{\Lambda}}_k^{t+1}\hat{\mathbf{\Lambda}}_k^{t+1^T} + \hat{\mathbf{\Psi}}_k^{t+1}$$

UNTIL: convergence criterion $|\mathcal{M}^{t+1} - \mathcal{M}^t| < \gamma$ is met

5.2 MPKL: A Model Selection Criterion

We define the maximum absolute pairwise difference between KL divergence values (MPKL) for a K-component GMM as follows:

$$MPKL = \max_{1 \le k_1, k_2 \le K} |KL(\mathcal{N}_{k_1}, \mathcal{N}_{k_2}) - KL(\mathcal{N}_{k_2}, \mathcal{N}_{k_1})| \tag{9}$$

As discussed earlier, it is an indicator of how well the separation between clusters is. This can be used as a criterion for selecting the number of clusters. For a chosen range of number of clusters $(2, \ldots, L)$, we compute MPKL for each value and choose K that minimizes MPKL:

$$\text{argmin}_{K \in [2,\ldots,L]} MPKL$$

5.3 Computational Complexity

The computational complexity is dominated by evaluating KLF and KLB which involves $O(K^2)$ matrix inversion steps ($O(p^3)$). Therefore, the overall computational complexity of all our algorithms (SIA,SIA-HD and model selection) is $O(K^2p^3)$. Appendix C.3 presents a comparison of runtime with EM and AD-GD.

5.4 An Upper Bound on the SIA Objective Function

Consider fitting a two-component GMM and without loss of generality, assume, $|\hat{\mathbf{\Sigma}}_1^t| \le |\hat{\mathbf{\Sigma}}_2^t|$, at some iteration t. We also

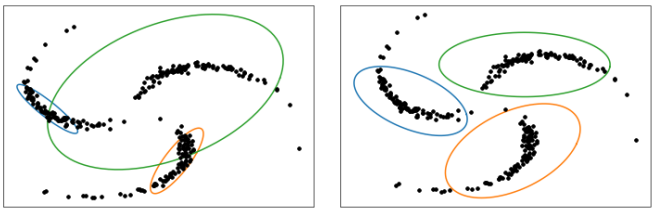
assume that the spectral norm of $\hat{\mathbf{\Sigma}}_1^t, \hat{\mathbf{\Sigma}}_2^t$ is bounded by some $c < \infty$ as a regularity condition. Let C denote a constant dependent only on p, n, w_1, w_2 and c and independent of $\boldsymbol{\theta}$. Let $\mathcal{M}$ denote the penalized log-likelihood (this result holds for both SIA and SIA-HD). We prove that the use of the penalty terms leads to a bounded objective function.

Theorem 1 *For non-negative weights w_1, w_2 , and any iteration t, $\mathcal{M} \leq -\frac{n}{2}(1+w_2)\left(\log(\frac{w_2}{1+w_2}\hat{\mathbf{\Sigma}}_2^t) + p\right) + C$.*

The proof is in Appendix E.1. Also note that the bound is given in terms of w_2 because we assume $|\hat{\mathbf{\Sigma}}_1^t| \leq |\hat{\mathbf{\Sigma}}_2^t|$. By setting $w_2 = 0$, we can see that the maximum of unpenalized likelihood is unbounded. If a $\mathbf{\Sigma}$ collapses, the log-likelihood may increase but the trace term and the determinant in the penalization will explode. Hence, by having negative weights $-w_1, -w_2$ we can ensure that $\mathcal{M}$ is bounded and control the behavior of the algorithm. A visualization of the likelihood surface, with and without penalty terms, is shown in Appendix F.

5.5 Illustrative Examples

We show how SIA improves the clustering performance in the three specific examples discussed in earlier sections. Fig. 6 shows the clustering obtained on the Pinwheel data discussed in section 3. We observe the difference in the clustering after the first and second steps of SIA. After the second step, compared to the clustering after step 1, the likelihood decreases, both the KLF and KLB values decrease, the ARI increases, and the clusters have less overlap.

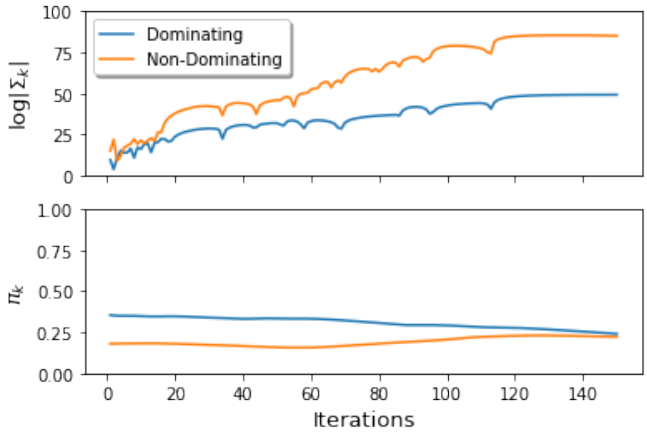


(a) After Step 1:
ARI = 0.65
KLDivs = {258,494},
Loglikelihood = -376

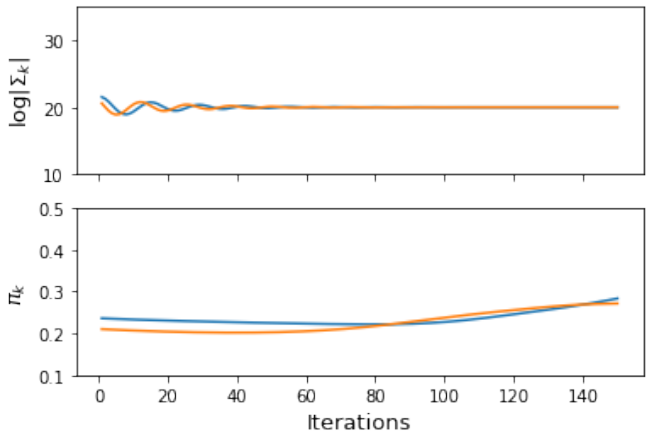
(b) After Step 2:
ARI = 0.96
KLDivs = {37,36},
Loglikelihood = -437

Fig. 6: Clustering using SIA in 3-component pinwheel data. Compare with figure 2.

On the high-dimensional dataset, from section 3 we find that SIA increases the variance of the non-dominating component (fig. 7 (a)) and the clustering performance remains low with ARI = 0.048. Running SIA-HD gives an ARI of 0.35; furthermore, there is no dominating component in the final clustering (fig. 7 (b)).



(a) SIA: ARI = 0.048
Loglikelihood = -19023

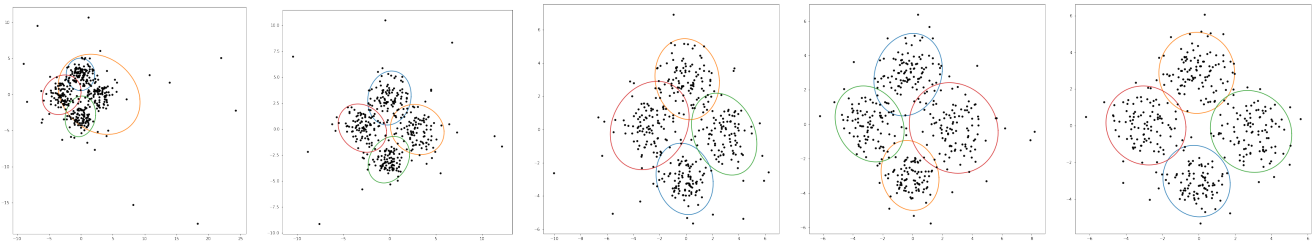


(b) SIA-HD: ARI = 0.35
Loglikelihood = -14807

Fig. 7: In HD ($p >> n$): (a) SIA: KLF and KLB are reduced by increasing the variance of non-dominating component in SIA, (b) SIA-HD: there is no dominating component in the final clustering.

Fig. 8 shows the clustering obtained by SIA on the contaminated mixture data from section 4. Table 5 shows the values of Loglikelihood (LL), ARI, MPKL, KLF and KLB before and after step 2 of SIA. The use of SIA decreases the likelihood but improves the clustering performance (ARI). The improvement in ARI is higher for the case with smaller degree of freedom, where misspecification is higher. A similar analysis shows the efficacy of SIA on datasets contaminated with random noise in Appendix C.2.

A limitation of SIA is that when cluster covariances are asymmetric in terms of their volume, its performance deteriorates. For a discussion on clustering such data we refer the reader to Aitkin and Wilson (1980); Jorgensen (1990).



(a) D.o.F = 2 (b) D.o.F = 3 (c) D.o.F = 4 (d) D.o.F = 5 (e) D.o.F = 10

Fig. 8: SIA improves the clustering performance when the Gaussian components are corrupted by a multivariate t-distribution - Refer to table 5

Table 5: Improvement in clustering performance before and after SIA step-2 for the contaminated case

	Before SIA Step 2					After SIA Step 2				
DoF	LL	KLF	KLB	MPKL	ARI	LL	KLF	KLB	MPKL	ARI
2	-1837.6	158.4	60.7	58.6	0.57	-1943.9	36.8	41.6	8.8	0.76
3	- 1813.4	57.2	80.9	16.6	0.37	-1871.0	48.2	46.2	4.3	0.79
4	-1725.8	48.6	89.1	11.7	0.75	-1735.1	39.6	54.3	3.8	0.78
5	-1683.9	67.1	82.9	11.12	0.79	-1694.5	46.1	54.6	4.9	0.82
10	- 1640.7	68.0	96.9	13.9	0.87	-1648.3	56.3	65.8	4.2	0.87

6 Simulation Studies

6.1 Misspecified Mixtures

We simulate non-Gaussian datasets with varying cluster separation and imbalance across component sizes. We evaluate the clustering performance of misspecified 3-component GMM using three different inference techniques – EM, AD-GD and our SIA algorithm.

Varying Cluster Separation

We simulate 50 datasets from a 2-dimensional mixture model with 3 Gaussian components, $\mathcal{N}((\lambda,\lambda),\mathbb{I}_2), \mathcal{N}((-\lambda,\lambda),\mathbb{I}_2)$, $\mathcal{N}((\lambda,-\lambda),\mathbb{I}_2)$, where λ is a scaling factor which controls the cluster separation and $\mathbb{I}_2$ is a unit covariance matrix. We evaluate the performance for λ values $\{3,4,5,7\}$. We sample 100 datapoints from each of these 3 components. These sampled datapoints are cubed so none of the components is normally distributed.

Table 6 shows the mean ARI, with standard deviation (SD), over these datasets, of the clusterin solutions obtained by the algorithms. We observe that only when there is high cluster separation ($\lambda = 7$), EM performs well. In the remaining three cases, of lower cluster separation, SIA outperforms AD-GD and EM.

Unbalanced mixtures

We simulate data from a 2 dimensional 2-component mixture. The mixture means $(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)$ are ((0.5,0),(-0.5,0)) and covariances matrices are $\boldsymbol{\Sigma}_k = (\boldsymbol{\Sigma}_k^{1/2}(\boldsymbol{\Sigma}_k^{1/2})^T)$. The parameters of $\boldsymbol{\Sigma}_k^{1/2}$ are sampled randomly to capture different covariance structures. The simulated datapoints are then cubed (for misspecification). Keeping the number of data points (N_1) in cluster 1 constant at 100, we vary the number of datapoints (N_2) in cluster 2 as $\{20, 50, 100\}$. We simulate 15 different datasets for each setting.

The mean ARI obtained by the methods are given in table 7. We observe that SIA outperforms AD-GD and EM in all cases. Appendix C.5 shows an illustration on a specific dataset.

Table 6: ARI (mean and SD) on varying λ

λ	3	4	5	7
SIA	**0.517**	**0.812**	**0.969**	0.994
	(0.091)	**(0.093)**	**(0.039)**	(0.007)
AD-GD	0.311	0.686	0.856	0.997
	(0.211)	(0.122)	(0.076)	(0.004)
EM	0.162	0.416	0.864	**1**
	(0.112)	(0.061)	(0.175)	**(0)**

Table 7: ARI (mean and SD) on varying imbalance

N_2	100	50	20
SIA	**0.123**	**0.195**	**0.224**
	(0.157)	**(0.247)**	**(0.286)**
AD-GD	0.101	0.148	0.151
	(0.180)	(0.239)	(0.275)
EM	0.101	0.150	0.196
	(0.130)	(0.221)	(0.282)

6.2 High-Dimensional Data

We compare the performance of EM and AD-GD (on both GMM and MFA) with that of SIA-HD, on high-dimensional datasets with varying n/p and covariance structures.

Varying $\mathbf{n/p}$

Following (Hoff, 2004, 2005; Pan and Shen, 2007; Wang and Zhu, 2008; Guo et al., 2010), we simulate 10 datasets with 100 datapoints from a 2-component GMM, with 50 datapoints from each component. In these datasets both the components have unit spherical covariance matrices but differ in their mean vectors. 90% of the dimensions of the mean vector are kept same (with value 0) across both the components and 10% of the dimensions of the mean vector differ (0 in one and 1 in the other cluster). Hence, for the purpose of clustering, only 10% of the dimensions are discriminating features and remaining 90% of the dimensions are noise features.

Table 8 shows the mean ARI obtained by SIA-HD, AD-GD and EM by fitting GMMs and MFA on high-dimensional datasets. EM fails in high dimensions while all the AD-GD algorithms run to completion. In the more challenging cases of low n/p ratios SIA-HD outperforms AD-GD for both GMM and MFA.

Table 8: ARI (mean and SD) on varying n/p

n/p	100/200	100/100	100/50	100/10
SIA-HD (GMM)	**0.801**	**0.517**	**0.333**	0.046
	(0.058)	**(0.131)**	**(0.168)**	(0.045)
AD-GD (GMM)	0.738	0.474	0.239	**0.046**
	(0.051)	(0.135)	(0.146)	**(0.040)**
SIA-HD (MFA)	0.753	0.330	0.250	0.015
	(0.086)	(0.114)	(0.180)	(0.030)
AD-GD (MFA)	0.523	0.275	0.124	0.018
	(0.119)	(0.069)	(0.113)	(0.031)
EM (GMM)	-	-	-	0.028
				(0.041)
EM (MFA)	-	-	0.098	0.006
			(0.128)	(0.023)

Varying Covariance

To evaluate the effects of various covariance structures, we sampled two p-dimensional mean vectors, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$, and two $p \times p$ matrices, $\boldsymbol{\Sigma}_1^{\frac{1}{2}}, \boldsymbol{\Sigma}_2^{\frac{1}{2}}$. Each element of the vectors $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are sampled independently from a standard normal distribution. In order to ensure the covariance matrix is PD, we multiply the sampled matrices with their respective transposes, i.e. $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_k^{\frac{1}{2}} \boldsymbol{\Sigma}_k^{\frac{1}{2}^T}$. We simulate 50 datasets for each case.

Table 9 shows that SIA-HD outperforms AD-GD, with the improvement increasing with decreasing n/p. Only the best performing baseline method (AD-GD) is shown.

Table 9: ARI (mean and SD) for varying covariance structures

n/p	100/200	100/100	100/50
SIA-HD (GMM)	**0.1825**	**0.256**	**0.108**
	(0.210)	**(0.241)**	**(0.140)**
AD-GD (GMM)	0.079	0.124	0.108
	(0.119)	(0.185)	(0.145)

Model Selection in High Dimensions

We follow the approach used in (Hoff, 2004, 2005; Pan and Shen, 2007; Wang and Zhu, 2008; Guo et al., 2010) and simulate a 4 component 50-dimensional dataset as shown in table 10. We assume unit spherical covariance matrices for each component. Further, the parameters of each component across the 50 dimensions are varied as shown in table 10, i.e., all the components have the same parameters for 35 dimensions, for component 2 only dimensions 1-5 are discriminating, for component 3 only dimensions 5-10 are discriminating and for component 4 only dimensions 10-15 are discriminating. The value of λ controls the cluster separation. For each component we sampled ten datapoints. Three different sets of simulated data with varying cluster separation are obtained by choosing the value of λ to be 1, 5 and 10 respectively. This experiment is repeated with 10 different seeds for each value of λ. We use SIA-HD for clustering with each value of $K \in \{3, 4, 5\}$. We compare the number of clusters selected using AIC and MPKL.

Table 10: Simulations for Model Selection

Features	C-1	C-2	C-3	C-4
1-5	$\mathcal{N}(0,1)$	$\mathcal{N}(\lambda,1)$	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$
5-10	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$	$\mathcal{N}(\lambda,1)$	$\mathcal{N}(0,1)$
10-15	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$	$\mathcal{N}(\lambda,1)$
15-50	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$	$\mathcal{N}(0,1)$

Table 11 shows that at very low cluster separation ($\lambda = 1$), both the criteria do not select 4 clusters. At moderate and high cluster separation ($\lambda = 5, 10$), BIC underestimates the number of clusters to 3, which is consistent with previous findings (Melnykov et al., 2010). AIC, on the other hand, performs well only when the cluster separation is very high. MPKL identifies 4 clusters in 7 out of 10 times at moderate separation and 8 out of 10 times at high cluster separation.

Table 11: Number of times 3, 4, and 5 components are selected using each criterion for a 4-component dataset.

λ	# clusters:	3	4	5
1	MPKL	10	0	0
1	AIC	10	0	0
1	BIC	10	0	0
5	MPKL	1	**7**	2
5	AIC	10	0	0
5	BIC	10	0	0
10	MPKL	1	**8**	1
5	AIC	0	10	0
10	BIC	10	0	0

7 Real Datasets

We compare the performance of SIA to AD-GD and EM on five real datasets – Wine (Forina et al., 1986), IRIS (Fisher, 1936), Abalone (Nash et al., 1994)), Urban Land cover (Johnson and Xie, 2013) and LUSC RNA-Seq (Kandoth et al., 2013)).

Table 12: Real datasets: Clustering performance (ARI) of SIA, EM and AD-GD

Dataset	n	p	K	SIA	EM	AD-GD
IRIS	150	4	3	0.922	0.903	0.903
Wine	178	13	3	0.570	0.462	0.375
Abalone	4177	8	3	0.128	0.121	0.089
Urban	168	148	9	0.112	-	0
LUSCRNAseq	130	206	2	0.0228	-	0

Table 12 shows the number of observations and dimensions in each dataset. Further, using the number of clusters and ground truth labels provided, we compare the ARI. We observe that SIA obtains the highest ARI in all the five datasets. In high-dimensions, EM fails to run and AD-GD assigns all the points to a single cluster corresponding to the dominating component. We now present detailed analysis of Wine and LUSCRNA-Seq datasets.

7.1 Wine dataset

The Wine dataset (Forina, 1991) contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. We use this dataset, without the labels of wine types, for analyzing the performance of the clustering algorithms.

We fit EM, AD-GD and SIA for $K \in \{2, 3, 4\}$ and report their log-likelihood (LL), MPKL, KLDivs, AIC and ARI values in table 13. For a given value of K we show

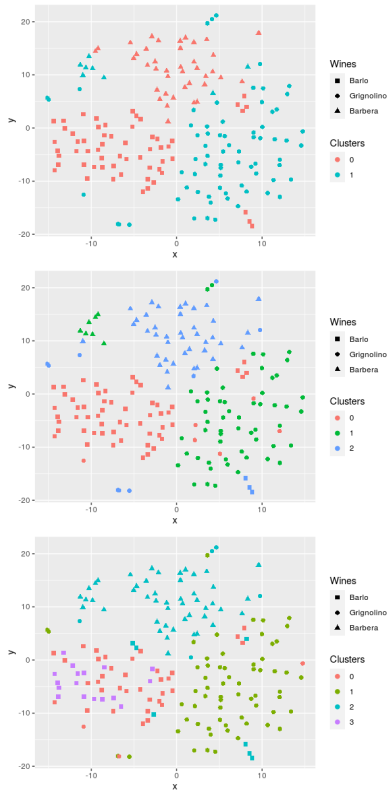


Fig. 9: tSNE plots for the best model for each value of K (S. Nos. 3,9 and 13 in Table 13); Clusters obtained for $K = 2$ (above), $K = 3$ (middle) and $K = 4$ (below).

the results for multiple runs of EM with different initializations to illustrate how loglikelihood along with our proposed MPKL heuristic can be used for model selection.

For $K = 2$, among the three solutions from EM - while clustering output (S. No. 1) has better LL of 2983.6, its MPKL of 46.8 is higher (worse) than those obtained by the other clustering outputs (S. No. 2 and S. No. 3). So, even among EM solutions, by trading off likelihood for an improvement in MPKL, we can choose better clustering solutions with higher accuracy. The LL of SIA is better than that of AD-GD (which was used in step 1 of SIA) indicating the KLDivs based penalization can aid in escaping local maxima. The ARI obtained by SIA (S. No. 5) is comparable to the best results from EM.

We observe similar trends for $K = 3$. Among EM solutions (S. No. 6 and 7), the solution with better MPKL (S. No. 7) achieves a better ARI. Again, the LL of SIA (S. No. 9) is better than that of AD-GD (S. No. 8 which was used in step 1 of SIA) indicating the KLDivs based penalization can escape local maxima. The best ARI is achieved by SIA which also has the lowest MPKL value.

For $K = 4$, we observe that AD-GD solution (S. No. 12) does not have high LL, nor does it have a low MPKL value. So, we initialize SIA step-2 with the output of EM;

Table 13: Clustering Outputs for Wine dataset

S. No.	K	Algorithm	LL	AIC	KLF	KLB	MPKL	ARI
1	2	EM	-2983.6	6385.3	86.3	39.5	46.8	0.46
2	2	EM	-3117.4	6652.9	17.2	14.8	2.3	0.57
3	2	EM	-3083.0	6584.0	22.2	17.36	4.8	0.59
4	2	AD-GD	-3081.6	6581.2	56.1	29.6	26.5	0.55
5	2	SIA	-3072.0	6562.0	42.1	27.5	14.6	0.57
6	3	EM	-2901.0	6430.0	321.29	91.44	229.8	0.46
7	3	EM	-2915.7	6459.4	93.25	118	51.68	0.618
8	3	AD-GD	-3030.04	6688.8	165.9	152.1	50.7	0.385
9	3	SIA	-2921.45	6470.9	57.7	60.8	**14.66**	**0.630**
10	4	EM	-2637.7	**6113.4**	28622214.3	407.7	13486984	0.52
11	4	EM	-2660.7	6158.0	5136379.7	362.1	5136017.5	0.52
12	4	AD-GD	-2910.7	6659.4	1192.4	364.0	486.65	0.3546
13	4	SIA	-2850.5	6539.0	519.0	304.9	214.05	**0.67**

Table 14: Wine data: Clustering results for the best model for each K (S. Nos. 3,9 and 13 in Table 13)

	$K=2$		$K=3$			$K=4$			
Cultivar	1	2	1	2	3	1	2	3	4
Barolo	59		56		3	35		7	17
Grignolino		71	6	56	9	3	63	5	
Barbera	40	8		7	41			48	

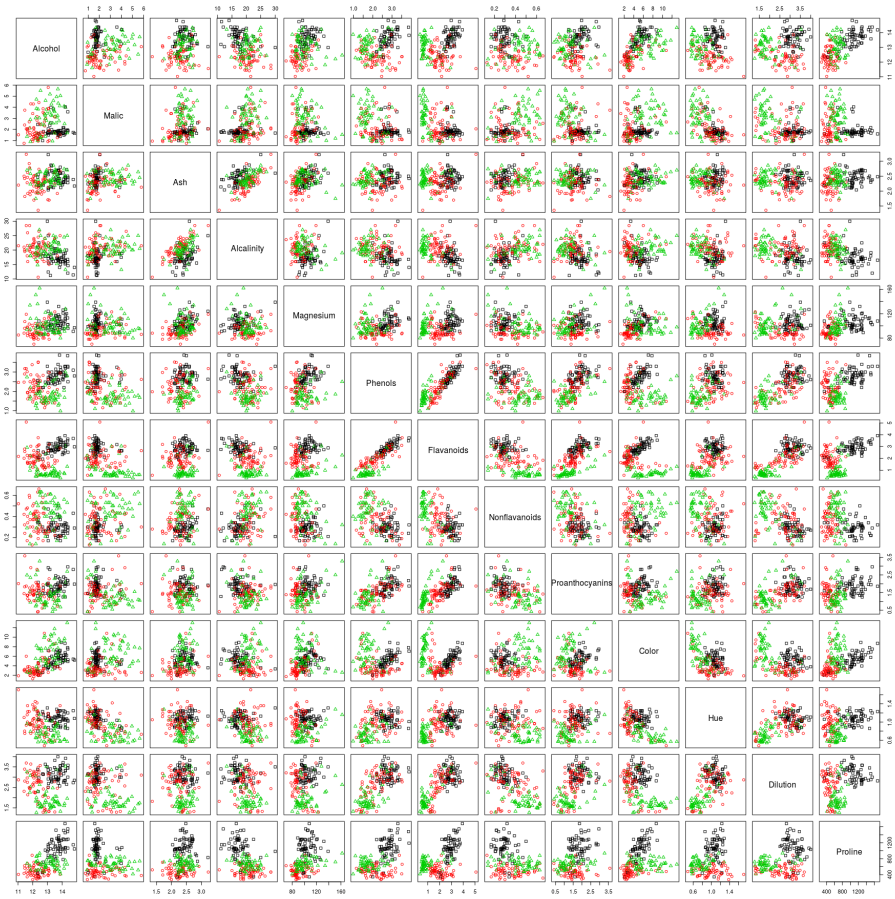


Fig. 10: Scatterplot matrix for the recommended model

we observe a decrease in MPKL and an improvement in the clustering performance (ARI) over both EM and AD-GD.

Given the low MPKL values and better AIC values, we recommend S. No. 9 as the best model for this dataset. We give the confusion matrices and tSNE plots (Maaten and Hinton, 2008) of the best (with respect to the MPKL heuristic) models for each K in Table 14 and in fig. 9 respectively. From the tSNE plots as well, we can see that $K = 3$ gives best and most well separated clusters. The scatterplot matrix for the recommended model is given in fig. 10.

7.2 Lung Cancer RNA-Seq Dataset

The LUSCRNA-seq data (Kandoth et al., 2013) contains gene expression and methylation data (206 attributes in total) for 130 patients with lung cancer as available from the TCGA database (Weinstein et al., 2013). Following Kasa et al. (2019), we use the (binary) survival status, which was not used as input for clustering, as the labels for computing ARI.

The MPKL criterion identifies 2 clusters as shown in Table 15. We compare the performance of SIA-HD with

Table 15: MPKL for LUSCRNA-seq dataset

# of clusters	2	3	4
MPKL	6.739	27.26	9.74

EM (MFA) and state-of-the-art model-based clustering algorithms for high-dimensional data:

- HDDC (Bouveyron and Brunet-Saumard, 2014), based on subspace clustering and parsimonious modeling.
- VarSelLCM (Marbac and Sedki, 2017), based on variable selection.
- MixGlasso (Städler et al., 2017), a penalized likelihood approach designed to automatically adapt to the sample size and scale of clusters.

Table 16: Clustering performance on LUSCRNA data

Algorithm	ARI	RI
SIA-HD (MFA)	0.0228	0.51
EM (MFA)	-	-
HDDC	-	-
MixGlasso	-0.01	0.49
VarSelLCM	-0.08	0.49

For all EM-based algorithms, 5 different K-Means initializations were used, where K-Means itself uses a random initialization. The result with the best AIC is reported. Table 16 shows the ARI and Rand Index (RI) obtained by all the methods. EM and HDDC fail to run. SIA-HD has the best ARI and RI.

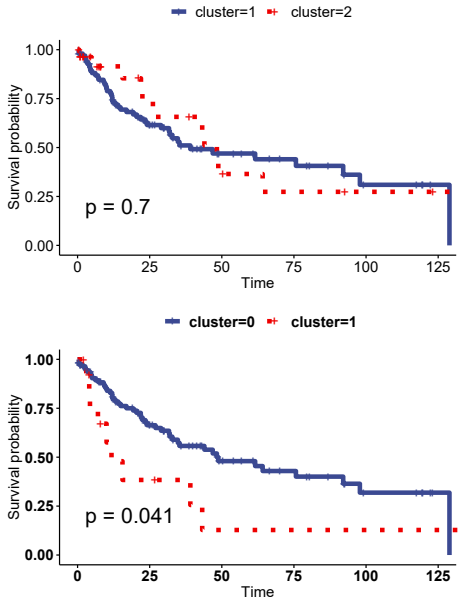
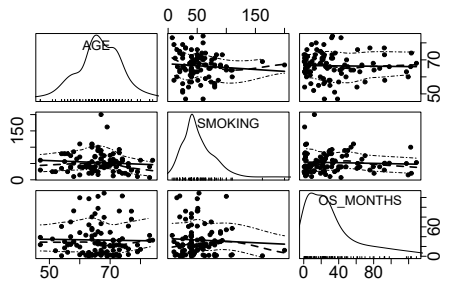
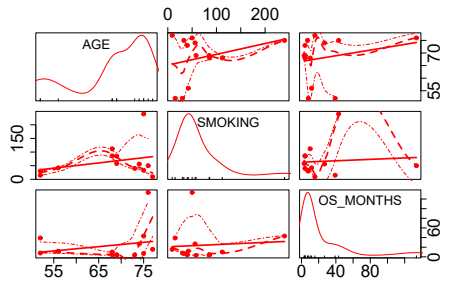


Fig. 11: Kaplan-Meier survival curves with p-values; Clusters obtained from MixGlasso (left) and SIA-HD (right).



(a) High Survival



(b) Low Survival

Fig. 12: Bivariate scatterplots of three features – age (in years), smoking (number of packs per year) and survival (in months) – for two clusters obtained from SIA-HD.

We evaluate the clusters obtained by SIA-HD and MixGlasso, with respect to their clinical subtypes, in terms of survival probability. The survival probabilities are estimated using Kaplan-Meier method and the p-values are obtained by the log-rank test on the survival analysis based on the cluster memberships. Fig. 11 shows the survival curves obtained on clusters from SIA-HD and MixGlasso. We observe that only SIA-HD yields significant clusters (p-value < 0.05) that are the least overlapping indicating a good separation of patient subtypes.

Fig. 12 shows bivariate dependencies in the two clusters from SIA-HD, thereby illustrating its ability to find meaningful patterns. Each scatterplot shows pairwise dependencies and the univariate distribution (diagonal). The bivariate pattern between smoking and survival is distinctly different between the two clusters: in the cluster with higher survival probability patients tend to smoke less when they are older.

8 Conclusion

In this paper we empirically investigated model-based clustering using EM and AD-GD in two practically relevant cases of misspecified GMMs and high-dimensional data. Empirically we found that EM has better clustering performance (in terms of ARI) in cases of misspecification while AD-GD outperforms EM on high-dimensional data. However, in misspecified cases, we observe inferior clustering solutions with poor ARI both from EM and AD. Such solutions have components with large variance and occur frequently with many different initializations. With high-dimensional data, we find that inferior solutions have components with low variance and high mixture weight.

To address these problems, we proposed a new penalty term based on the Kullback Leibler divergence between pairs of fitted components. We developed algorithms SIA, SIA-HD for clustering and the MPKL criterion for model selection. Gradient computations for this penalized likelihood is difficult but AD-GD based optimization, that does not require closed-form derivatives, can be done effortlessly. Experiments on synthetic and real datasets demonstrate the efficacy of this penalized likelihood for clustering and model selection – in cases of misspecification and high-dimensional data. Software implementation of our algorithm and the experiments conducted are available online[2].

Our experiments also illustrate the benefits of automatic differentiation (AD), that is well known but rarely used, for parameter inference in mixture models. We compared AD-GD with EM for two cases – GMM with unconstrained covariances and MFA. The flexible framework of AD-GD and our KL divergence based penalty term can be used for many families of mixture models, e.g., PGMM, as constraints imposed by these models can be easily added to AD-GD. A limitation of SIA is that when cluster covariances are asymmetric in terms of their volume, its performance deteriorates. Future work can address this limitation and explore the use of our general KL divergence based penalty term for non-Gaussian mixtures and our model selection criterion for other models. Theoretical properties such as convergence and sample complexity can also be investigated in the future.

[2] `https://bitbucket.org/cdal/sia/`

Acknowledgements V.R. was supported by Singapore Ministry of Education Academic Research Fund [R-253-000-138-133].

References

Aitkin M, Wilson GT (1980) Mixture models, outliers, and the EM algorithm. Technometrics 22(3):325–331

Akaho S, Kappen HJ (2000) Nonmonotonic generalization bias of Gaussian mixture models. Neural Computation 12(6):1411–1427

Alexandrovich G (2014) An exact Newton's method for ML estimation of a Gaussian mixture

Amari SI, Park H, Ozeki T (2006) Singularities affect dynamics of learning in neuromanifolds. Neural Computation 18(5):1007–1065

Atkinson SE (1992) The performance of standard and hybrid EM algorithms for ML estimates of the normal mixture model with censoring. Journal of Statistical Computation and Simulation 44(1-2):105–115

Banfield JD, Raftery AE (1993) Model-based Gaussian and non-Gaussian clustering. Biometrics pp 803–821

Baydin AG, Pearlmutter BA, Radul AA, Siskind JM (2018) Automatic differentiation in machine learning: a survey. Journal of Machine Learning Research 18:1–43

Biernacki C (2004) An asymptotic upper bound of the likelihood to prevent gaussian mixtures from degenerating. Preprint, Université de Franche-Comté, Besançon

Biernacki C, Celeux G, Govaert G (2003) Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. Computational Statistics & Data Analysis 41(3-4):561–575

Bouveyron C, Brunet-Saumard C (2014) Model-based clustering of high-dimensional data: A review. Computational Statistics & Data Analysis 71:52–78

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press

Cai TT, Ma J, Zhang L, et al. (2019) Chime: Clustering of high-dimensional Gaussian mixtures with EM algorithm and its optimality. The Annals of Statistics 47(3):1234–1267

Carpenter B, Hoffman MD, Brubaker M, Lee D, Li P, Betancourt M (2015) The Stan math library: Reverse-

mode automatic differentiation in C++. arXiv preprint arXiv:150907164

Chen J, Tan X (2009) Inference for multivariate normal mixtures. Journal of Multivariate Analysis 100(7):1367–1383

Cuesta-Albertos JA, Gordaliza A, Matrán C, et al. (1997) Trimmed k-means: An attempt to robustify quantizers. The Annals of Statistics 25(2):553–576

Day NE (1969) Estimating the components of a mixture of normal distributions. Biometrika 56(3):463–474

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B (Methodological) 39(1):1–22

Dwivedi R, Khamaru K, Wainwright MJ, Jordan MI, et al. (2018) Theoretical guarantees for EM under misspecified Gaussian mixture models. In: Advances in Neural Information Processing Systems, pp 9681–9689

Farcomeni A, Greco L (2016) Robust methods for data reduction. CRC press

Fisher RA (1936) The use of multiple measurements in taxonomic problems. Annals of Eugenics 7(2):179–188

Fop M, Murphy TB, et al. (2018) Variable selection methods for model-based clustering. Statistics Surveys 12:18–65

Forina M (1991) UCI machine learning repository wine dataset. https://archive ics uci edu/ml/datasets/Wine

Forina M, Armanino C, Castino M, Ubigli M (1986) Multivariate data analysis as a discriminating method of the origin of wines. Vitis 25:189–201

Fournier DA, Skaug HJ, Ancheta J, Ianelli J, Magnusson A, Maunder MN, Nielsen A, Sibert J (2012) AD model builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optimization Methods and Software 27(2):233–249

García-Escudero LA, Gordaliza A, Matrán C, Mayo-Iscar A (2010) A review of robust clustering methods. Advances in Data Analysis and Classification 4(2-3):89–109

García-Escudero LA, Gordaliza A, Mayo-Íscar A (2014) A constrained robust proposal for mixture modeling avoiding spurious solutions. Advances in Data Analysis and Classification 8(1):27–43

García-Escudero LA, Gordaliza A, Greselin F, Ingrassia S, Mayo-Iscar A (2016) The joint role of trimming and constraints in robust estimation for mixtures of gaussian factor analyzers. Computational Statistics & Data Analysis 99:131–147

Ghahramani Z, Hilton G (1997) The EM algorithm for mixture of factor analyzers. Techical Report CRG-TR-96-1

Giraud C (2014) Introduction to high-dimensional statistics. Chapman and Hall/CRC

Greselin F, Ingrassia S (2015) Maximum likelihood estimation in constrained parameter spaces for mixtures of factor analyzers. Statistics and Computing 25(2):215–226

Guo J, Levina E, Michailidis G, Zhu J (2010) Pairwise variable selection for high-dimensional model-based clustering. Biometrics 66(3):793–804

Hathaway R (1985) A constrained formulation of maximum-likelihood estimation for normal mixture distributions. The Annals of Statistics 13:795–800

Hoff P (2004) Discussion of "Clustering objects on subsets of attributes", by Friedman and Meulman. Journal of the Royal Statistical Society, Series B 66(4):845

Hoff P (2005) Subset clustering of binary sequences, with an application to genomic abnormality data. Biometrics 61(4):1027–1036

Hubert L, Arabie P (1985) Comparing partitions. Journal of Classification 2(1):193–218

Ingrassia S (2004) A likelihood-based constrained algorithm for multivariate normal mixture models. Statistical Methods and Applications 13(2):151–166

Ingrassia S, Rocci R (2007) Constrained monotone EM algorithms for finite mixture of multivariate Gaussians. Computational Statistics & Data Analysis 51:5339–5351

Ingrassia S, Rocci R (2011) Degeneracy of the EM algorithm for the mle of multivariate Gaussian mixtures and dynamic constraints. Computational Statistics & Data Analysis 55(4):1715–1725

Iwata T, Duvenaud D, Ghahramani Z (2013) Warped mixtures for nonparametric cluster shapes. In: Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, AUAI Press, pp 311–320

Jamshidian M, Jennrich RI (1997) Acceleration of the EM algorithm by using quasi-Newton methods. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 59(3):569–587

Johnson B, Xie Z (2013) Classifying a high resolution image of an urban area using super-object information. ISPRS journal of photogrammetry and remote sensing 83:40–49

Jorgensen MA (1990) Influence-based diagnostics for finite mixture models. Biometrics pp 1047–1058

Kandoth C, McLellan MD, Vandin F, Ye K, Niu B, Lu C, Xie M, Zhang Q, McMichael JF, Wyczalkowski MA, et al. (2013) Mutational landscape and significance across 12 major cancer types. Nature 502(7471):333

Kasa SR, Bhattacharya S, Rajan V (2019) Gaussian mixture copulas for high-dimensional clustering and dependency-based subtyping. Bioinformatics

Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR) 1412

Kucukelbir A, Tran D, Ranganath R, Gelman A, Blei DM (2017) Automatic differentiation variational inference. The Journal of Machine Learning Research 18(1):430–474

Li H, Xu Z, Taylor G, Studer C, Goldstein T (2018) Visualizing the loss landscape of neural nets. In: Advances in

Neural Information Processing Systems, pp 6389–6399
Maaten Lvd, Hinton G (2008) Visualizing data using t-sne. Journal of Machine Learning Research 9(Nov):2579–2605
Maclaurin D, Duvenaud D, Adams RP (2015) Autograd: Effortless gradients in numpy. In: ICML 2015 AutoML Workshop, vol 238
Marbac M, Sedki M (2017) Variable selection for model-based clustering using the integrated complete-data likelihood. Statistics and Computing 27(4):1049–1063
Margossian CC (2019) A review of automatic differentiation and its efficient implementation. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9(4):e1305
McLachlan G, Krishnan T (2007) The EM algorithm and extensions, vol 382. John Wiley & Sons
McLachlan GJ, Peel D (2000) Finite Mixture Models. John Wiley & Sons, New York
McNicholas P, Murphy T (2008) Parsimonious Gaussian mixture models. Statistics and Computing 18:285–296
Melnykov V, Maitra R, et al. (2010) Finite mixture models and model-based clustering. Statistics Surveys 4:80–116
Nash WJ, Sellers TL, Talbot SR, Cawthorn AJ, Ford WB (1994) The population biology of abalone (haliotis species) in tasmania. i. blacklip abalone (h. rubra) from the north coast and islands of bass strait. Sea Fisheries Division, Technical Report 48:p411
Pan W, Shen X (2007) Penalized model-based clustering with application to variable selection. Journal of Machine Learning Research 8(May):1145–1164
Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic differentiation in PyTorch
Pav SE (2016) Madness: a package for multivariate automatic differentiation
Peel D, McLachlan G (2000) Robust mixture modelling using the t distribution. Statistics and Computing 10:339–348
Punzo A, McNicholas PD (2016) Parsimonious mixtures of multivariate contaminated normal distributions. Biometrical Journal 58(6):1506–1537
Redner RA, Walker HF (1984) Mixture densities, maximum likelihood and the EM algorithm. SIAM review 26(2):195–239
Ritter G (2014) Robust cluster analysis and variable selection. CRC Press
Robert C (2014) Machine learning, a probabilistic perspective. Taylor & Francis
Ruwet C, Garcia-Escudero LA, Gordaliza A, Mayo-Iscar A (2013) On the breakdown behavior of the tclust clustering procedure. Test 22(3):466–487
Salakhutdinov R, Roweis ST, Ghahramani Z (2003) Optimization with EM and expectation-conjugate-gradient. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03), pp 672–679
Skaug HJ (2002) Automatic differentiation to facilitate maximum likelihood estimation in nonlinear random effects models. Journal of Computational and Graphical Statistics 11(2):458–470
Skaug HJ, Fournier DA (2006) Automatic approximation of the marginal likelihood in non-Gaussian hierarchical models. Computational Statistics & Data Analysis 51(2):699–709
Städler N, Dondelinger F, Hill SM, Akbani R, Lu Y, Mills GB, Mukherjee S (2017) Molecular heterogeneity at the network level: high-dimensional testing, clustering and a TCGA case study. Bioinformatics 33(18):2890–2896
Wang S, Zhu J (2008) Variable selection for model-based high-dimensional clustering and its application to microarray data. Biometrics 64(2):440–448
Weinstein JN, Collisson EA, Mills GB, Shaw KRM, Ozenberger BA, Ellrott K, Shmulevich I, Sander C, Stuart JM, Network CGAR, et al. (2013) The cancer genome atlas pan-cancer analysis project. Nature Genetics 45(10):1113
White H (1982) Maximum likelihood estimation of misspecified models. Econometrica: Journal of the Econometric Society pp 1–25
Xu L, Jordan MI (1996) On convergence properties of the EM algorithm for Gaussian mixtures. Neural Computation 8(1):129–151
Zhou H, Pan W, Shen X (2009) Penalized model-based clustering with unconstrained covariance matrices. Electronic Journal of Statistics 3:1473

A Symbols and Notation

Symbol	Meaning
f	overall probability density from which data is sampled
g	misspecified model for data sampled from f
f_i	probability density of the i-th component
π_i	mixing proportions or weight of the i-th component
$\boldsymbol{\theta}_i$	parameters of i-th mixture component
$\boldsymbol{\theta}$	all parameters of the model
$\boldsymbol{\theta}^*$	true parameters of the model from which data is sampled
$\boldsymbol{\Theta}$	set of all possible parameters i.e. the entire parameter space
$\mathbf{F}_i$	factors associated with i-th datapoint
K	total of number of components in the model
$k, k^{'}, k_1, k_2$	indices over the total number of components K
$\mathbf{X}_i$	i-th random datapoint
$\mathbf{x}_i$	i-th observed datapoint
$\mathbf{X}$	all the n random datapoints
$\mathbf{x}$	all the n observed datapoints
n	Total number of iid datapoints considered
p	Total number of dimensions or features of the data
p_e	Total number of free parameters in the model
$\mathcal{L}(\boldsymbol{\theta})$	Log-Likelihood of observing $\mathbf{x}$ if the parameters were $\boldsymbol{\theta}$
$\mathcal{N}$	density of multi-variate Gaussian distribution
$\boldsymbol{\mu}_i, \mu_{ij}$	mean of i-th component of a GMM and its value of along j-th dimension respectively
$\boldsymbol{\Sigma}_i, \Sigma_{ijk}$	Covariance matrix of i-th component of a GMM and its jk-th element respectively
$\mathbf{U}_i$	square-root of i-th covariance matrix $\boldsymbol{\Sigma}_i$
$\hat{a}^t$	Estimate of parameter a at the end of iteration t
c, C	Constants independent of model parameters
λ	factor controlling the cluster separation in simulations
λ_k	hyperparameter controlling the fitted component covariance
$\mathbb{I}$	Identity matrix
ϵ	learning rate in the vanilla gradient descent step
γ	convergence threshold for stopping criterion

Table 17: Symbols used in the paper

B Automatic Differentiation

The examples in this section are from Baydin et al. (2018). In symbolic differentiation, we first evaluate the complete expression and then differentiate it using rules of differential calculus. First note that a naive computation may repeatedly evaluate the same expression multiple times, e.g., consider the rules:

$$\frac{d}{dx}\left(F(x)+G(x)\right) \rightsquigarrow \frac{d}{dx}F(x)+\frac{d}{dx}G(x) \tag{10}$$

$$\frac{d}{dx}\left(F(x)\,g(x)\right) \rightsquigarrow \left(\frac{d}{dx}F(x)\right)G(x)+F(x)\left(\frac{d}{dx}G(x)\right) \tag{11}$$

Let $H(x) = F(x)G(x)$. Note that $H(x)$ and $\frac{d}{dx}H(x)$ have in common: $F(x)$ and $G(x)$, and on the right hand side, $F(x)$ and $\frac{d}{dx}F(x)$ appear separately. In symbolic differentiation we plug the derivative of $F(x)$ and thus have nested duplications of any computation that appears in common between $F(x)$ and $\frac{d}{dx}F(x)$. In this manner symbolic differentiation can produce exponentially large symbolic expressions which take correspondingly long to evaluate. This problem is called **expression swell**. To illustrate the problem, consider the following iterations of the logistic map $l_{n+1} = 4l_n(1-l_n)$, $l_1 = x$ and the corresponding derivatives of l_n with respect to x. Table 18 clearly shows that the number of repetitive evaluations increase with n.

If the symbolic form is not required and only numerical evaluation of derivatives is required, computations can be simplified by storing the values of intermediate sub-expressions. Further efficiency gains in computation can be achieved by interleaving differentiation and simplification steps. The derivative of $l_{n+1} = 4l_n(1-l_n)$ can be found using the chain rule $\frac{dl_{n+1}}{dl_n}\frac{dl_n}{dl_{n-1}}\dots\frac{dl_1}{dl_x}$ which simplifies to $4(1-l_n-l_n)4(1-l_{n-1}-l_{n-1})\dots4(1-x-x)$. Note that evaluation in AD is computationally linear in n (because we add only one $(1-l_n-l_n)$ for each increase by 1). This linear time complexity is achieved due to 'carry-over' of the derivatives at each step, rather than evaluating the derivative at the end and substituting the value of x.

Table 18: Iterations of the logistic map $l_{n+1} = 4l_n(1 - l_n)$, $l_1 = x$ and the corresponding derivatives of l_n with respect to x, illustrating expression swell (from Baydin et al. (2018))

n	l_n	$\frac{d}{dx}l_n$	$\frac{d}{dx}l_n$ (Simplified form)
1	x	1	1
2	$4x(1-x)$	$4(1-x)-4x$	$4-8x$
3	$16x(1-x)(1-2x)^2$	$16(1-x)(1-2x)^2 - 16x(1-2x)^2 - 64x(1-x)(1-2x)$	$16(1-10x+24x^2-16x^3)$
4	$64x(1-x)(1-2x)^2\,(1-8x+8x^2)^2$	$128x(1-x)(-8+16x)(1-2x)^2(1-8x+8x^2) + 64(1-x)(1-2x)^2(1-8x+8x^2)^2 - 64x(1-2x)^2(1-8x+8x^2)^2 - 256x(1-x)(1-2x)(1-8x+8x^2)^2$	$64(1-42x+504x^2-2640x^3+7040x^4-9984x^5+7168x^6-2048x^7)$

The Python code below shows the simplicity of the implementation for this problem.

```
from autograd import grad
def my_func(x,n):
    p = x
    y = x * (1 - x)
    for i in range(n):
        y = y*(1 - y)

    return y
grad_func = grad(my_func)
grad_func(0.5,4)
```

Consider the following recursive expressions: $l_0 = \frac{1}{1+e^x}$, $l_1 = \frac{1}{1+e^{l_0}}$, ..., $l_n = \frac{1}{1+e^{l_{n-1}}}$ We evaluate the derivative of l_n with respect to x and compare the runtime in Mathematica (SD) vs PyTorch (AD) for various values of n. As n increases, it is expected that runtime also increases. It can be seen from the results in Table 19 that runtime increases linearly for AD (using PyTorch) whereas it increases exponentially for SD (using Mathematica).

Table 19: Average runtime (over 1000 runs)

n	AD	SD
1	0.00013	0.00000
5	0.00030	0.00005
10	0.00051	0.00023
50	0.00293	0.00437
100	0.00433	0.15625
200	0.00917	1.45364

C Additional Experimental Results

C.1 EM and AD-GD: A comparison of clustering performance

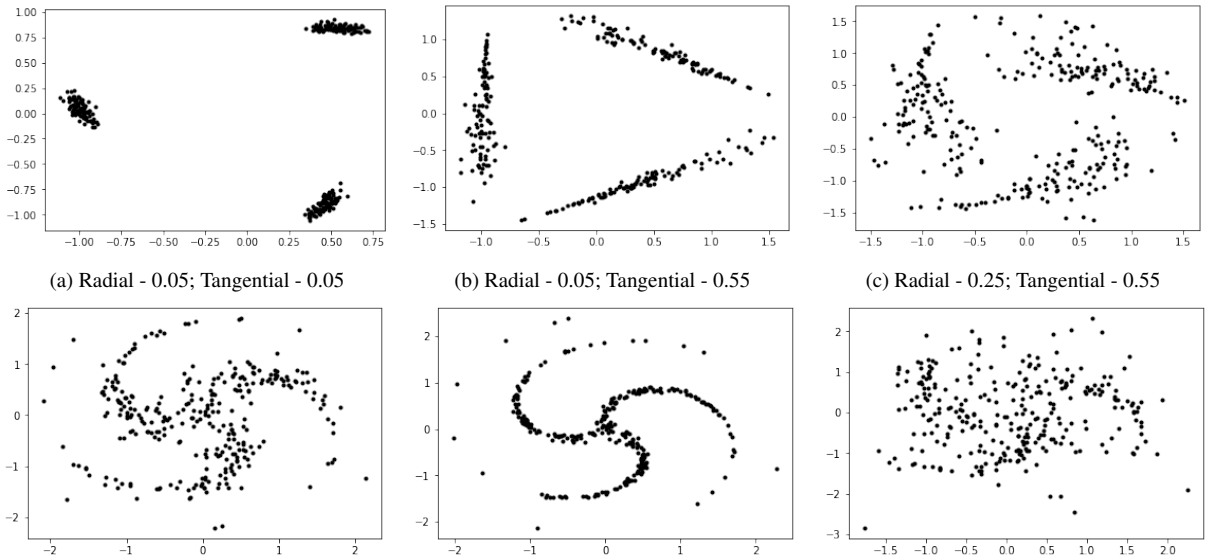
We compare the clustering performance of AD-GD and EM on pure GMM data as well as non-GMM (misspecified) data. We do so by simulating 3600 datasets as described below and running both the algorithms with matching settings with respect to random initialization, number of iterations (maximum of 100 iterations) and convergence threshold (1e-5). The details of the simulations and results are given below.

Pure GMM data : We simulate datasets from 36 different 2–dimensional 3–component Gaussian Mixture Models that differ in their means and covariance matrices. This is to test different types of data - from well-separated components to highly overlapping components. The GMM parameters are varied as follows:

- First, we construct a 2×1 vector $\mathbf{v}_k$ and a 2×2 matrix $\mathbf{Z}_k$ for each of these 3 components, where each element of the vectors and matrices are sampled from a standard normal distribution.
- The mean vector of the kth component is obtained by multiplying $\mathbf{v}_k$ by a factor $k_{\boldsymbol{\mu}}$ i.e. $\boldsymbol{\mu}_k = k_{\boldsymbol{\mu}}\mathbf{v}_k$. This factor $k_{\boldsymbol{\mu}}$ is varied from 0.25 to 1.5 in steps of 0.25, i.e., the factor $k_{\boldsymbol{\mu}}$ takes one of the six values in $\{0.25, 0.5, 0.75, 1, 1.25, 1.5\}$. Higher the value of $k_{\boldsymbol{\mu}}$, more the separation between the means of components.
- The $\mathbf{Z}_k$ vector is added to the 2×2 Identity matrix $\mathbb{I}$ (This step ensures the PD of the covariance matrix that is computed in the later steps).

– We multiply the matrix $\mathbf{Z}_k + \mathbb{I}$ by $k_{\boldsymbol{\Sigma}}$ to obtain $\mathbf{U}_k$. The scaling factor $k_{\boldsymbol{\Sigma}}$ is varied from 0.05 to 0.65 in steps of 0.1, i.e., the scaling factor takes one of the six values in $\{0.05, 0.15, 0.25, 0.35, 0.45, 0.55\}$. The covariance matrix of the kth component is $\boldsymbol{\Sigma}_k = \mathbf{U}_k \mathbf{U}_k^T$. Higher the scaling factor $k_{\boldsymbol{\Sigma}}$, more the inter-component overlap.

By choosing six different values of $k_{\boldsymbol{\mu}}$ and $k_{\boldsymbol{\Sigma}}$ each, we simulate 36 (6x6) GMM parameters with varying cluster separation. For each these 36 GMM parameters, we simulate 50 different datasets each containing 300 samples, for a total of 1800 datasets. Both the methods (EM and AD-GD) are run on each dataset with 50 random initializations and average ARI is computed for each combination of $(k_{\boldsymbol{\mu}}, k_{\boldsymbol{\Sigma}})$.



(a) Radial - 0.05; Tangential - 0.05 (b) Radial - 0.05; Tangential - 0.55 (c) Radial - 0.25; Tangential - 0.55

(d) Radial - 0.55; Tangential - 0.25 (e) Radial - 0.55; Tangential - 0.05 (f) Radial - 0.55; Tangential - 0.55

Fig. 13: Data with completely different shapes can be simulated by varying the tangential and radial parameters of pinwheel dataset.

Misspecified datasets: The pinwheel data is generated by sampling from Gaussian distributions and then stretching and rotating the data in a controlled manner. The centers are equidistant around the unit circle. The variance is controlled by two parameters r and t, the radial standard deviation and the tangential standard deviation respectively. The warping is controlled by a third parameter, s, the rate parameter. A datapoint (x, y) belonging to component or arm k is generated as follows:

$$(x', y') \sim \mathcal{N}(0, \mathbb{I}) \tag{12}$$

$$(x, y) = ((rx' + 1)\cos\theta_k + ty'\sin\theta_k, -(rx' + 1)\sin\theta_k + ty'\cos\theta_k), \tag{13}$$

where $\theta_k = k\dfrac{2\pi}{K} + s \times e^{rx'+1}$

The MATLAB [3] and Python[4] codes for generating pinwheel datasets are available online. We simulated 36 different pinwheel configurations by varying the radial and tangential components for 3-component mixture models using the Autograd package in Python. The rate parameter s of warping is fixed at 0.4 for all the 36 different combinations. This is to test multiple combinations – clusters with heavy tails to warped mixtures as shown in figure 13. The parameters of pinwheel datasets are chosen as follows:

– The radial parameter r is chosen to be one of the six values $\{0.05, 0.15, 0.25, 0.35, 0.45, 0.55\}$
– The tangent parameter t is chosen to be one of the six values $\{0.05, 0.15, 0.25, 0.35, 0.45, 0.55\}$

By choosing different values of r and t, we simulate 36 (6x6) set of pinwheel parameters. For each these 36 pinwheel parameters, we simulate 50 different datasets each with 300 samples, for a total of 1800 datasets. Please refer to figure 13 to see the various kinds of component shapes and separations that have been generated.

The tables below show the difference in average ARI between clusterings obtained by EM and those from AD-GD on GMM (table 20) and Pinwheel (table 21) datasets. The remaining tables (Tables 22 –29) show the mean and standard deviations over the GMM and Pinwheel datasets for both EM and AD-GD. From the results in tables 20 and 21, we observe that in both the cases (misspecification and nomisspecification), EM inference outperforms AG-GD inference. Alexandrovich (2014)'s work on exact Newton's method using analytical derivatives also report similar results for non misspecified cases.

[3] https://github.com/duvenaud/warped-mixtures/blob/master/data/pinwheel.m
[4] https://github.com/HIPS/autograd/blob/master/examples/data.py

Table 20: Difference in average ARI of GMM datasets

$k_{\boldsymbol{\mu}} - k_{\boldsymbol{\Sigma}}$	0.05	0.15	0.25	0.35	0.45	0.55
0.25	0.56	0.52	0.44	0.40	0.37	0.36
0.5	0.43	0.55	0.49	0.44	0.41	0.40
0.75	0.36	0.51	0.52	0.48	0.46	0.42
1	0.36	0.48	0.55	0.53	0.50	0.50
1.25	0.30	0.47	0.55	0.58	0.54	0.54
1.5	0.33	0.51	0.58	0.60	0.61	0.56

Table 21: Difference in average ARI Pinwheel datasets

$r - t$	0.05	0.15	0.25	0.35	0.45	0.55
0.05	0.18	0.30	0.41	0.50	0.51	0.08
0.15	0.47	0.50	0.56	0.58	0.57	0.24
0.25	0.51	0.58	0.61	0.62	0.61	0.47
0.35	0.30	0.39	0.45	0.53	0.49	0.43
0.45	0.12	0.17	0.20	0.24	0.27	0.30
0.55	0.05	0.08	0.12	0.16	0.18	0.19

Table 22: Average ARI GMM datasets using EM

$k_{\boldsymbol{\mu}} - k_{\boldsymbol{\Sigma}}$	0.05	0.15	0.25	0.35	0.45	0.55
0.25	0.78	0.63	0.52	0.46	0.42	0.41
0.5	0.81	0.79	0.68	0.59	0.53	0.51
0.75	0.84	0.84	0.78	0.70	0.64	0.57
1	0.84	0.85	0.83	0.77	0.70	0.67
1.25	0.84	0.86	0.84	0.82	0.75	0.71
1.5	0.83	0.86	0.86	0.83	0.81	0.74

Table 23: Average ARI of GMM datasets using AD-GD

$k_{\boldsymbol{\mu}} - k_{\boldsymbol{\Sigma}}$	0.05	0.15	0.25	0.35	0.45	0.55
0.25	0.23	0.11	0.08	0.06	0.06	0.05
0.5	0.38	0.24	0.18	0.15	0.12	0.11
0.75	0.47	0.33	0.26	0.21	0.18	0.16
1	0.48	0.37	0.28	0.24	0.20	0.17
1.25	0.54	0.39	0.29	0.24	0.21	0.17
1.5	0.50	0.35	0.28	0.23	0.20	0.18

Table 24: Average ARI of Pinwheel datasets using EM

$r - t$	0.05	0.15	0.25	0.35	0.45	0.55
0.05	0.95	1.00	1.00	1.00	0.95	0.48
0.15	0.95	0.97	1.00	0.98	0.93	0.58
0.25	0.88	0.95	0.95	0.94	0.91	0.74
0.35	0.60	0.69	0.73	0.78	0.72	0.64
0.45	0.36	0.41	0.41	0.44	0.45	0.46
0.55	0.23	0.26	0.28	0.31	0.32	0.32

Table 25: Average ARI Pinwheel datasets using AD-GD

$r - t$	0.05	0.15	0.25	0.35	0.45	0.55
0.05	0.77	0.70	0.59	0.50	0.44	0.40
0.15	0.48	0.47	0.44	0.40	0.36	0.34
0.25	0.37	0.37	0.34	0.32	0.29	0.27
0.35	0.30	0.30	0.28	0.25	0.23	0.21
0.45	0.24	0.24	0.22	0.20	0.18	0.16
0.55	0.19	0.18	0.17	0.15	0.14	0.12

Table 26: Std. Deviation of ARI GMM datasets using EM

$k_{\boldsymbol{\mu}} - k_{\boldsymbol{\Sigma}}$	0.05	0.15	0.25	0.35	0.45	0.55
0.25	0.30	0.27	0.25	0.24	0.22	0.23
0.5	0.29	0.25	0.26	0.27	0.26	0.25
0.75	0.26	0.23	0.25	0.26	0.26	0.27
1	0.25	0.23	0.22	0.26	0.26	0.26
1.25	0.25	0.24	0.23	0.22	0.26	0.26
1.5	0.24	0.23	0.22	0.23	0.22	0.26

Table 27: Std. Deviation of ARI of GMM datasets using AD-GD

$k_{\boldsymbol{\mu}} - k_{\boldsymbol{\Sigma}}$	0.05	0.15	0.25	0.35	0.45	0.55
0.25	0.21	0.13	0.09	0.08	0.07	0.05
0.5	0.26	0.21	0.17	0.15	0.12	0.10
0.75	0.25	0.23	0.21	0.19	0.16	0.14
1	0.28	0.23	0.21	0.19	0.17	0.15
1.25	0.29	0.24	0.21	0.18	0.16	0.15
1.5	0.27	0.25	0.23	0.20	0.18	0.16

Table 28: Std. Deviation of ARI Pinwheel datasets using EM

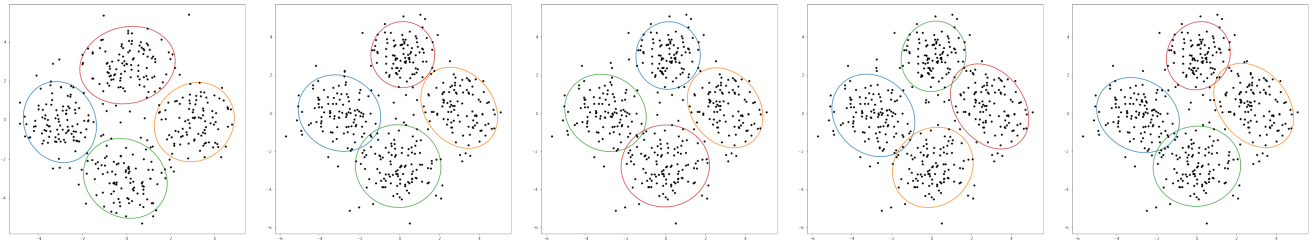
$r - t$	0.05	0.15	0.25	0.35	0.45	0.55
0.05	0.15	0.00	0.00	0.00	0.16	0.32
0.15	0.15	0.12	0.00	0.08	0.15	0.34
0.25	0.12	0.10	0.10	0.08	0.06	0.25
0.35	0.15	0.16	0.17	0.13	0.18	0.17
0.45	0.13	0.12	0.16	0.19	0.18	0.16
0.55	0.08	0.08	0.09	0.11	0.12	0.11

Table 29: Std. Deviation of ARI of Pinwheel datasets using AD-GD

$r - t$	0.05	0.15	0.25	0.35	0.45	0.55
0.05	0.03	0.04	0.04	0.04	0.05	0.05
0.15	0.04	0.04	0.05	0.04	0.05	0.04
0.25	0.04	0.04	0.04	0.04	0.04	0.04
0.35	0.03	0.03	0.04	0.04	0.03	0.03
0.45	0.03	0.03	0.03	0.03	0.03	0.03
0.55	0.03	0.03	0.03	0.03	0.03	0.03

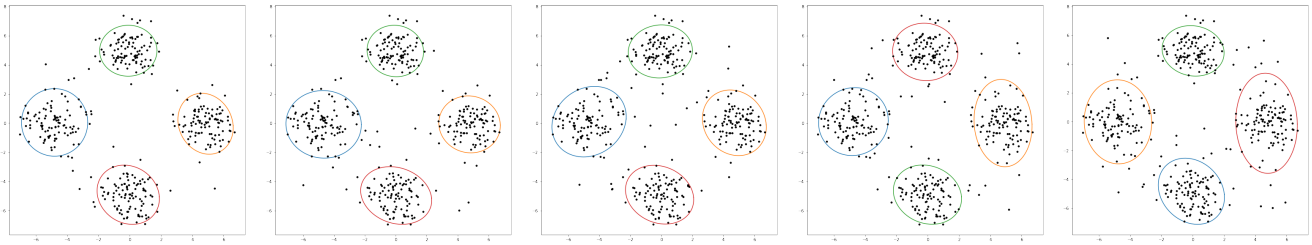
C.2 Noise

We construct a four component GMM with unit covariances and means $\{(-3,0),(0,3),(0,-3),(3,0)\}$. We sample 50 datapoints from each component. We add datapoints sampled uniformly from the square $(-6,6)\times(-6,6)$ as noise. We fit a GMM using EM in each case. Fig. 14 shows that as we increase the number of noisy datapoints added to the our original data sampled from GMM, the adverse effect of misspecification on EM increases. Fig. 16 shows that SIA performs well and obtains good clustering. Table 30 shows the loglikelihood, KLF, KLB, MPKL and ARI values for each case, before and after step 2 of SIA. We repeat the same experiment of adding noisy data samples, but with increased cluster separation by choosing the means as $\{(-5,0),(0,5),(0,-5),(5,0)\}$. Fig. 15 shows that increasing the cluster separation mitigates the effect of misspecification with EM.



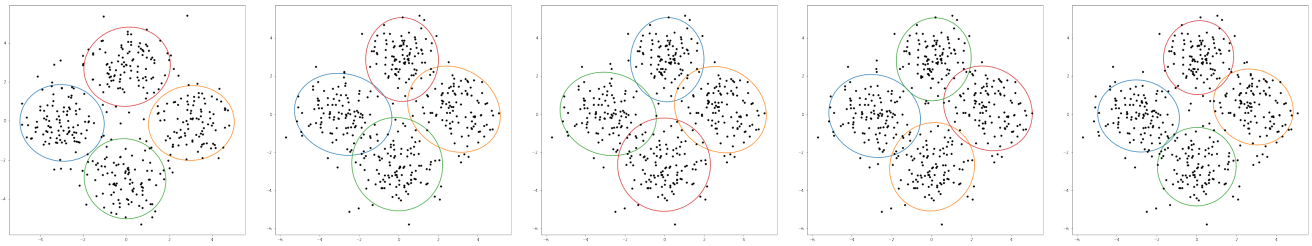
(a) # of noise samples = 10 (b) # of noise samples = 20 (c) # of noise samples = 30 (d) # of noise samples = 40 (e) # of noise samples = 50

Fig. 14: As the number of noisy samples increases, the affect of misspecification also increases



(a) # of noise samples = 10 (b) # of noise samples = 20 (c) # of noise samples = 30 (d) # of noise samples = 40 (e) # of noise samples = 50

Fig. 15: Increasing the cluster separation mitigates the effect of uniform noise on specification



(a) # of noise samples = 10 (b) # of noise samples = 20 (c) # of noise samples = 30 (d) # of noise samples = 40 (e) # of noise samples = 50

Fig. 16: SIA improves the clustering performance in the uniform noise case

Table 30: Improvement in clustering performance before and after SIA step-2 with increasing noisy samples $(10, \ldots, 50)$

	Before SIA					After SIA				
# of samples	LL	KLF	KLB	MPKL	ARI	LL	KLF	KLB	MPKL	ARI
10	-1676.9	75.6	85.2	4.4	0.91	-1681.3	69.0	70.6	1.2	0.92
20	-1719.6	85.2	68.2	8.4	0.87	-1737.8	51.3	45.0	2.4	0.89
30	-1761.1	64.0	87.2	6.9	0.88	-1778.4	44.2	52.8	2.3	0.89
40	-1806.2	78.1	67.3	5.6	0.87	-1815.9	55.2	51.7	2.2	0.89
50	-1839.6	79.4	67.1	6.2	0.87	-1845.5	70.2	65.7	2.0	0.88

C.3 Wallclock time

We benchmark the runtime of SIA with respect to AD-GD and EM. We do so by simulating 50 datasets, each containing 100 datapoints sampled randomly, for each setting. We run all the algorithms to convergence in likelihood (tolerance 1e-5) or a maximum of 50 iterations, which ever happens earlier. We evaluate with the number of clusters $K = \{3, 5\}$ and data dimensions $p = \{2, 5, 10, 50\}$. All experiments were run using Autograd, Numpy and Sklearn packages in Python 3.7 on Dell Windows 10 machine (Intel i7-6700 quadcore CPU@3.40GHz; 8 GB RAM; 500 GB HDD).

Table 31 shows the runtime of all the algorithms. We observed that with increasing dimensionality, the number of iterations for EM reduced, and at $p = 50$, EM failed to go beyond iteration 1. As expected, SIA takes roughly double the time taken by AD-GD.

Table 31: Average Runtime in seconds with data dimensions $2, 5, 10, 50$.

# of clusters	Algorithm	2	5	10	50
3	SIA	7.900	7.926	8.037	11.508
3	AD-GD	4.471	4.467	4.529	6.50
3	EM	0.009	0.006	0.009	-
5	SIA	20.571	20.785	21.238	30.041
5	AD-GD	11.127	11.194	11.433	16.167
5	EM	0.036	0.025	0.014	-

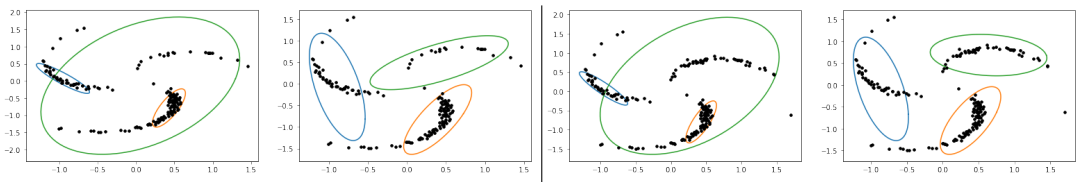
C.4 Summary statistics of solutions in sets 1–4 for EM

Table 32: Summary statistics of EM clustering solutions over 100 random initializations on the Pinwheel dataset (shown in fig. 2), grouped into 4 sets based on AIC ranges. Mean and standard deviation of AIC, ARI, component weights and covariance determinants, over solutions in each set.

Set	AIC Range	AIC	ARI	π_1	π_2	π_3	$\lvert\mathbf{\Sigma}_1\rvert$	$\lvert\mathbf{\Sigma}_2\rvert$	$\lvert\mathbf{\Sigma}_3\rvert$
1	771-773	771.9	0.625	0.257	0.265	0.477	0.0002	0.0005	0.125
		(0.0005)	(3e-16)	(5e-4)	(7e-5)	(1e-4)	(4e-7)	(1e-7)	(1e-4)
2	781-782	781.1	0.912	0.306	0.341	0.352	7e-4	0.01	0.01
		(2e-6)	(0)	(4e-6)	(3e-6)	(7e-6)	(1e-7)	(4e-7)	(2e-6)
3	786-787	786.8	0.652	0.257	0.266	0.475	2e-4	5e-4	0.157
		(0.001)	(0.002)	(5e-5)	(4e-4)	(4e-4)	2e-7	(6e-6)	(5e-4)
4	788-850	810.7	0.840	0.287	0.319	0.393	3e-3	0.04	0.84
		(10.84)	(0.084)	(0.022)	(0.009)	(0.03)	(1e-3)	(0.02)	(0.084)

C.5 Illustration of unbalanced clusters on Pinwheel data

We illustrate the effect of unbalanced clusters on the same pinwheel data that is discussed in section 3.2. In the first experiment, instead of (100,100,100) number of points respectively in the three clusters, we have (100,50,20) number of points respectively for the three clusters. Please refer to fig. 17 (a) and (b) for the clustering results with SIA with this dataset. In the second experiment we have (100,50,50) number of points. Please refer to fig. 17 (c) and (d) for the results obtained from SIA.



(a) After Step 1: ARI = 0.602 MPKL = 220.887 Loglikelihood = -139.4

(b) After Step 2: ARI = 0.804 MPKL = 41.77, Loglikelihood = -191.9

(c) After Step 1: ARI = 0.606 MPKL = 279.769, Loglikelihood = -228.2

(d) After Step 2: ARI =0.847 MPKL = 39.251, Loglikelihood = -261.52

Fig. 17: Clustering using SIA for unbalanced pinwheel data. Componentwise number of datapoints: $\{100, 50, 20\}$ (a and b), $\{100, 50, 50\}$ (c and d).

D Closed Forms

In this section, we illustrate the difficulty of deriving the closed form expressions of the gradients of the penalized likelihood used in SIA. These closed form expressions are required to develop an EM algorithm for inference. Consider the mean update in EM for the unpenalized loglikelihood:

$$\frac{\partial \log \mathcal{L}}{\partial \boldsymbol{\mu}_k} = -\sum_{i=1}^{n} \gamma(z_{ik}) \mathbf{\Sigma}_k^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k) \tag{14}$$

where $\gamma(z_{ik})$ is defined as the responsibility and is equal to $\frac{\pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_j, \mathbf{\Sigma}_j)}$. Setting $\frac{\partial \log \mathcal{L}}{\partial \boldsymbol{\mu}_k}$ to zero, we obtain $\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} \gamma(z_{ik}) \mathbf{x}_i}{\sum_{i=1}^{n} \gamma(z_{ik})}$. Note that the computation of the mean update $\boldsymbol{\mu}_k$ can be easily parallelized in the M-step because it does not involve other mean terms $\boldsymbol{\mu}_j$ where $j \neq k$.

Now, consider the gradient of the penalized loglikelihood:

$$\frac{\partial \log(\mathcal{L} - w_1 \times KLF - w_2 \times KLB)}{\partial \boldsymbol{\mu}_k} = \\ -\sum_{i=1}^{n} \gamma(z_{ik}) \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k) - w_1 \sum_{j \neq k} 2\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_j) - w_2 \sum_{j \neq k} 2\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_j) \tag{15}$$

Setting the above expression to zero, we can obtain the expression for the mean $\boldsymbol{\mu}_k$ as:

$$\boldsymbol{\mu}_k = \left(\sum_{i=1}^{n} \gamma(z_{ik}) \boldsymbol{\Sigma}_k^{-1} - (w_1 + w_2)(\sum_{j \neq k} \left(\boldsymbol{\Sigma}_j^{-1} + \boldsymbol{\Sigma}_k^{-1} \right)) \right)^{-1} \\ \left(\sum_{i=1}^{n} \gamma(z_{ik}) \boldsymbol{\Sigma}_k^{-1} \mathbf{x}_i - (w_1 + w_2)(\sum_{j \neq k} \left(\boldsymbol{\Sigma}_j^{-1} + \boldsymbol{\Sigma}_k^{-1} \right) \boldsymbol{\mu}_j) \right) \tag{16}$$

As seen above, even though the process of obtaining the updates is straightforward, it is laborious. For the update of covariance estimates, we could not find a closed form estimate for the penalized loglikelihood. Further, note the mean update requires knowledge of other mean terms $\boldsymbol{\mu}_j$ and hence cannot be parallelized. EM based updates for the more flexible models such as MFA will be even more cumbersome.

In contrast, using Gradient Descent (GD) based inference using Automatic Differentiation (AD) avoids the need to compute such laborious gradient updates by hand or to hard code them in the software program. AD-GD based inference approach is model-agnostic as AD software are blackbox tools where we just need to input the loglikelihood; they don't require closed forms of the gradients. Thus the same software implementation can be easily extended for more flexible models such as MFA, PGMM etc. This is not the case with the traditional EM based inference where the updates have to be derived and re-implemented for each model.

E Theoretical Results

E.1 Proof of Theorem 1

First, we show the boundedness for the simple case of a univariate mixture model as it is straightforward to follow and later proceed to the proof of the multivariate case. Throughout the proof we omit the superscript 't' on the estimates for ease of mathematical exposition.

Proof **Univariate:** With loss of generality, let the component 1 - $\mathcal{N}(x; \mu_1, \sigma_1^2)$ - is the one that is fitting over a single point and reducing its variance $\hat{\sigma}_1^2 \to 0$; also, assume that $\hat{\sigma}_1^2 \leq \hat{\sigma}_2^2 \leq c < \infty$

First note that the value of likelihood satisfies the following inequality:

$$\mathcal{N}(x; \mu, \sigma^2) \leq \frac{1}{\sqrt{2\pi}\sigma} \tag{17}$$

For one-dimensional case, the penalized likelihood of SIA1 for an observation x_i can be written as follows:

$$\log(\pi_1 \mathcal{N}(x_i; \mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(x_i; \mu_2, \sigma_2^2)) - w_1 \left(\log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \right) - w_2 \left(\log \frac{\sigma_1}{\sigma_2} + \frac{\sigma_2^2 + (\mu_1 - \mu_2)^2}{2\sigma_1^2} - \frac{1}{2} \right) \tag{18}$$

$$\leq \log(\frac{\pi_1}{\sigma_1} + \frac{\pi_2}{\sigma_2}) - w_2 \log \sigma_1 + \frac{w_3}{2\sigma_1^2} \tag{19}$$

$$\leq \log(\frac{\pi_1 + \pi_2}{\sigma_1}) - w_2 \log \sigma_1 + \frac{w_3}{2\sigma_1^2} \tag{20}$$

$$\log(\hat{\pi}_1\mathcal{N}(x_i;\hat{\mu}_1,\hat{\sigma}_1^2)+\hat{\pi}_2\mathcal{N}(x_i;\hat{\mu}_2,\hat{\sigma}_2^2)) - w_1\left(\log\frac{\hat{\sigma}_2}{\hat{\sigma}_1}+\frac{\hat{\sigma}_1^2+(\hat{\mu}_1-\hat{\mu}_2)^2}{2\hat{\sigma}_2^2}-\frac{1}{2}\right) - w_2\left(\log\frac{\hat{\sigma}_1}{\hat{\sigma}_2}+\frac{\hat{\sigma}_2^2+(\hat{\mu}_1-\hat{\mu}_2)^2}{2\hat{\sigma}_1^2}-\frac{1}{2}\right) \tag{21}$$

(By using eq 17 and setting $(w_3 = w_2(\hat{\sigma}_2^2+(\hat{\mu}_1-\hat{\mu}_2)^2))$, and leaving out -ve terms)

$$\leq \log(\frac{\hat{\pi}_1}{\hat{\sigma}_1}+\frac{\hat{\pi}_2}{\hat{\sigma}_2}) - w_2\log\hat{\sigma}_1 - \frac{w_3}{2\hat{\sigma}_1^2} - \log\sqrt{2\pi} + w_2\log c \tag{22}$$

$$\leq \log(\frac{\hat{\pi}_1+\hat{\pi}_2}{\hat{\sigma}_1}) - w_2\log\hat{\sigma}_1 - \frac{w_3}{2\hat{\sigma}_1^2} + w_2\log c (\text{where } w_4 = 1+w_2)$$

$$\leq -w_4\log\hat{\sigma}_1 - \frac{w_3}{2\hat{\sigma}_1^2} + w_2\log c \tag{23}$$

(Differentiating wrt σ_1 and setting to 0, we find $\sigma_1^* = \sqrt{\frac{w_3}{w_4}}$)

$$\leq -w_4\log\sqrt{\frac{w_3}{w_4}} - 0.5w_4 + w_2\log c \tag{24}$$

The terms in equation 24 are all constants hence the SIA 1 likelihood is bounded.

Multivariate: $\lambda_1(\hat{\Sigma}_1), \lambda_1(\hat{\Sigma}_2) \leq c_1$ where $\lambda_p(.)$ and $\lambda_1(.)$ denotes the smallest and the largest eigenvalue of the $\mathbb{R}^{p\times p}$ matrix. The likelihood can be made unbounded by making the determinant of $\hat{\Sigma}_1$ close to zero i.e. $|\hat{\Sigma}_1| \rightarrow 0$.

First, we bound the following terms:

$$0.5w_2\log(|\hat{\Sigma}_2|) \leq 0.5w_2p\log(c_1) = C(\text{some constant}) \tag{25}$$

$$\log(\hat{\pi}_1\mathcal{N}(x_i;\hat{\mu}_1,\hat{\Sigma}_1)+\hat{\pi}_2\mathcal{N}(x_i;\hat{\mu}_2,\hat{\Sigma}_2)) - 0.5*w_1\left(\log\frac{|\hat{\Sigma}_2|}{|\hat{\Sigma}_1|}+\text{tr}(\hat{\Sigma}_2^{-1}\hat{\Sigma}_1)+(\mu_2-\mu_1)^T\hat{\Sigma}_2^{-1}(\mu_2-\mu_1)^T\right)$$
$$-0.5*w_2\left(\log\frac{|\hat{\Sigma}_1|}{|\hat{\Sigma}_2|}+\text{tr}(\hat{\Sigma}_1^{-1}\hat{\Sigma}_2)+(\mu_2-\mu_1)^T\hat{\Sigma}_1^{-1}(\mu_2-\mu_1)^T\right) \tag{26}$$

(By using eq 17 and 25, and leaving out -ve terms and absorbing the constants into C)

$$\leq 0.5*\log(\frac{\hat{\pi}_1}{|\hat{\Sigma}_1|}+\frac{\hat{\pi}_2}{|\hat{\Sigma}_2|}) - 0.5*w_2\log(|\hat{\Sigma}_1|) - 0.5*w_2\text{tr}(\hat{\Sigma}_1^{-1}\hat{\Sigma}_2) \tag{27}$$

$$= 0.5*\left(\log(\frac{\hat{\pi}_1+\hat{\pi}_2}{|\hat{\Sigma}_1|}) - w_2\log(|\hat{\Sigma}_1|) - w_2\text{tr}(\hat{\Sigma}_1^{-1}\hat{\Sigma}_2)\right) \tag{28}$$

$$= 0.5*\left(-\log(|\hat{\Sigma}_1|) - w_2\log(|\hat{\Sigma}_1|) - w_2\text{tr}(\hat{\Sigma}_1^{-1}\hat{\Sigma}_2)\right) \tag{29}$$

$$= 0.5*\left(-w_3\log(|\hat{\Sigma}_1|) - w_2\text{tr}(\hat{\Sigma}_1^{-1}\hat{\Sigma}_2)\right) \quad (\text{where } w_3 = 1+w_2) \tag{30}$$

$$= 0.5*\left(w_3\log(|\hat{\Sigma}_1^{-1}|) - w_2\text{tr}(\hat{\Sigma}_1^{-1}\hat{\Sigma}_2)\right) \tag{31}$$

Note that equation 31 is concave in Σ_1^{-1} as $\log(|.|)$ is a concave function and trace is affine (Boyd and Vandenberghe, 2004); hence, if a maximum exists, it is the global maximum. We can find the maximum by differentiating with respect to Σ_1^{-1} and setting to zero;

$$\nabla_{\Sigma_1^{-1}}\left(w_3\log(|\hat{\Sigma}_1^{-1}|) - w_2\text{tr}(\hat{\Sigma}_1^{-1}\hat{\Sigma}_2)\right) = w_3\hat{\Sigma}_1 - w_2\hat{\Sigma}_2 = 0 \tag{32}$$

$$\implies \hat{\Sigma}_1 = \frac{w_2}{w_3}\hat{\Sigma}_2 \tag{33}$$

Therefore, the final upper bound of the log-likelihood is given by

$$-0.5(1+w_2)\left(\log(\frac{w_2}{1+w_2}\hat{\Sigma}_2)+p\right)+C \tag{34}$$

F Likelihood Surface Visualization

To understand how the penalization in (7) helps in clustering we visualize the log-likelihood surface using the technique of Li et al. (2018) for neural network loss landscapes. Given two sets of parameters $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$, the surface function $S(\alpha,\beta) = \mathcal{L}(\alpha\boldsymbol{\theta}_1 + \beta\boldsymbol{\theta}_2)$ where $0 \leq \alpha, \beta \leq 1$ is used to analyze the log-likelihood $\mathcal{L}$.

As an illustration, we use the Pinwheel data described in section 3.2 with the parameters obtained for (g) and (a) of fig. 2 as $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ respectively and plot the log-likelihood and penalized log-likelihood in fig. 18 (a) and (b) respectively. It is evident in fig. 18 (a) that the log-likelihood has a significant plateau region which has almost similar (unpenalized) likelihood. However, for SIA likelihood in (b), such a plateau does not exist and there exists a clear maxima at $(\alpha = 1, \beta = 0)$. As expected $\boldsymbol{\theta}_1$ ((g) of fig. 2) has a better clustering output, despite having similar (unpenalized) likelihood as that of $\boldsymbol{\theta}_2$.

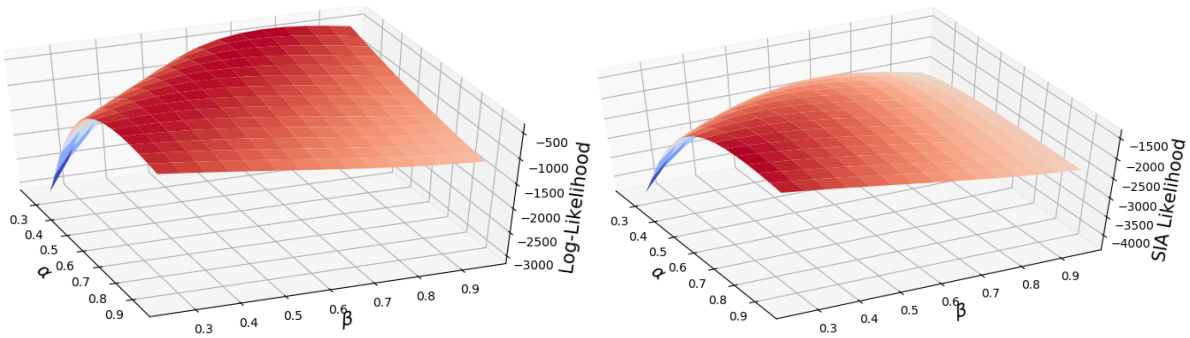


Fig. 18: (Left) Log-likelihood shows a plateau region ; (Right) Penalized log-likelihood has no plateau and a clear peak corresponding to good clustering.