An Evolutionary Algorithm for Matrix-Variate Model-Based Clustering

AN EVOLUTIONARY ALGORITHM FOR MATRIX-VARIATE MODEL-BASED CLUSTERING

BY

THOMAS J. FLYNN, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF MATHEMATICS & STATISTICS

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

© Copyright by Thomas J. Flynn, September 2023

All Rights Reserved

Master of Science (2023) McMaster University

(Mathematics & Statistics) Hamilton, Ontario, Canada

TITLE: An Evolutionary Algorithm for Matrix-Variate Model-Based Clustering

AUTHOR: Thomas J. Flynn

B.Sc., (Mathematics and Statistics)

McMaster University, Hamilton, Ontario Canada

SUPERVISOR: Dr. Paul D. McNicholas

NUMBER OF PAGES: ix, 37

To my family and McKayla.

Abstract

Model-based clustering is the use of finite mixture models to identify underlying group structures in data. Estimating parameters for mixture models is notoriously difficult, with the expectation-maximization (EM) algorithm being the predominant method. An alternative approach is the evolutionary algorithm (EA) which emulates natural selection on a population of candidate solutions. By leveraging a fitness function and genetic operators like crossover and mutation, EAs offer a distinct way to search the likelihood surface. EAs have been developed for model-based clustering in the multivariate setting; however, there is a growing interest in matrix-variate distributions for three-way data applications. In this context, we propose an EA for finite mixtures of matrix-variate distributions.

Acknowledgements

First I would like thank my supervisor, Dr. Paul McNicholas, for his guidance into the world of model-based clustering.

I would also like to thank Dr. Noah Forman and Dr. Ben Bolker for serving as members of my examining committee.

Lastly, I would like to thank my friends and family for their endless support over my 6-year journey at McMaser University. I could not have done it without them.

Contents

Abstract **iv**

Acknowledgements **v**

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**

2 Background **3**

2.1 Matrix-Variate Distributions 3

2.2 Supervised, Unsupervised, and Semi-Supervised Learning 4

2.3 Model-Based Clustering 5

2.3.1 Historical Perspective 5

2.3.2 Finite Mixture Models 6

2.3.3 Parameter Estimation 7

2.3.4 Convergence Criterion 9

2.3.5 Model Selection 10

2.3.6 Model Performance 11

2.3.7 Benefits Over Vectorization 12
2.4 Evolutionary Computation 12
2.4.1 Evolutionary Algorithms 12
2.4.2 Genetic Operators 13
2.4.3 Evolutionary Algorithms For Model-Based Clustering 14

3 Methods **15**
3.1 Model and Fitness Function 15
3.2 Algorithm Design 16
3.3 Procedure 20

4 Simulation **21**
4.1 Simulation 1 22
4.2 Simulation 2 23

5 Application **25**
5.1 Landsat Satellite Dataset 25
5.2 Handwritten Digits Dataset 29

6 Conclusions **31**
6.1 Discussion 31
6.2 Future Work 32

Bibliography **33**

List of Tables

4.1 Mean and standard deviation of ARI, runtime, and likelihood ratio for EA and EM associated with Simulation 1. 23

4.2 Mean and standard deviation of ARI, runtime, and likelihood ratio for EA and EM associated with Simulation 2. 24

5.1 Description of the Landsat test set classes and observation count. . . 26

5.2 Converged log-likelihood, ARI, runtime, and EA to EM likelihood ratio associated with the Landsat dataset. 28

5.3 Cross-tabulation of the EA MAP classifications associated with the Landsat dataset. 28

5.4 Cross-tabulation of the EM MAP classifications associated with the Landsat dataset. 28

5.5 Converged log-likelihood, ARI, runtime, and EA to EM likelihood ratio associated with the handwritten digits dataset. 30

5.6 Cross-tabulation of the EA MAP classifications associated with the handwritten digits dataset. 30

5.7 Cross-tabulation of the EM MAP classifications associated with the handwritten digits dataset. 30

List of Figures

3.1 Illustration of the crossover operator, randomly swapping distinct rows $\tilde{\mathbf{z}}_2$ and $\tilde{\mathbf{z}}_N$ from the parent solution to create a child with similar genetic material. 17

3.2 Illustration of the mutation operator, swapping unit-element $\tilde{z}_{2G}$ with zero-element $\tilde{z}_{22}$ in a random row of a surviving parent for a slightly modified solution. 18

Chapter 1

Introduction

Model-based clustering is the use of finite mixture models to identify underlying group structures in data. A recent review of model-based clustering can be found in McNicholas (2016b), while extensive details are available in McNicholas (2016a). Estimating parameters for mixture models is notoriously difficult, with the expectation-maximization (EM) algorithm being the predominant method (Dempster *et al.*, 1977). Although the EM algorithm is a powerful technique, it is susceptible to becoming trapped at local maxima, which translates to suboptimal clustering results (Titterington *et al.*, 1985). An alternative approach is the evolutionary algorithm (EA) which emulates natural selection on a population of candidate solutions. By leveraging a fitness function and genetic operators like crossover and mutation, EAs offer a distinct way to search the likelihood surface, helping to bypass the limitations of the EM algorithm.

While EAs have been successful for multivariate model-based clustering (Andrews and McNicholas, 2013; McNicholas *et al.*, 2020), there remains a paucity of research in the matrix-variate setting. Matrix-variate or three-way data consists of matrices

organized in three dimensions. This structure commonly arises in multivariate longitudinal data, with multiple measurements taken at different time points, and greyscale image data. Model-based clustering has demonstrated its effectiveness in clustering three-way data by leveraging mixtures of matrix-variate distributions (Viroli, 2011; Anderlucci and Viroli, 2015; Doğru *et al.*, 2016; Gallaugher and McNicholas, 2018; Silva *et al.*, 2023).

In response to the growing interest in matrix-variate data applications, this work develops an EA for matrix-variate model-based clustering. The algorithm incorporates both crossover and mutation operations and is applied in the matrix-variate normal setting. We find that its performance is competitive against the EM algorithm for clustering three-way data in both simulated and real-world datasets.

Chapter 2

Background

The following sections provide essential context for this research, covering matrix variate distributions, types of machine learning algorithms, clustering, and evolutionary computation. Clustering is presented through the model-based paradigm including discussions on finite mixtures of matrix-variate distributions, parameter estimation, convergence criterion, model selection, and model performance. Evolutionary computation is presented with a focus on evolutionary algorithms, the genetic operators crossover and mutation, and their application to model-based clustering.

2.1 Matrix-Variate Distributions

Three-way data refers to datasets that consist of matrices organized in three dimensions. These datasets are characterized by having n units (rows), p variables (columns), and N occasions (layers). Matrix-variate distributions provide an effective way to model three-way data, the most mathematically tractable being the matrix-variate normal distribution (Gupta and Nagar, 1999). An $n \times p$ random matrix $\mathscr{X}$

follows an $n \times p$ matrix-variate normal distribution, denoted $\mathcal{N}_{n\times p}(\mathbf{M}, \mathbf{\Sigma}, \mathbf{\Psi})$, if its density can be written as

$$\phi_{n\times p}(\mathbf{X}|\mathbf{M}, \mathbf{\Sigma}, \mathbf{\Psi}) = \frac{1}{(2\pi)^{\frac{np}{2}}|\mathbf{\Sigma}|^{\frac{p}{2}}|\mathbf{\Psi}|^{\frac{n}{2}}} \exp\left\{-\frac{1}{2}\text{tr}\left(\mathbf{\Sigma}^{-1}(\mathbf{X}-\mathbf{M})\mathbf{\Psi}^{-1}(\mathbf{X}-\mathbf{M})'\right)\right\},$$

where $\mathbf{M}$ is the $n \times p$ location matrix, $\mathbf{\Sigma}$ is the $n \times n$ row covariance matrix, and $\mathbf{\Psi}$ is the $p \times p$ column covariance matrix. An equivalent formulation in the multivariate setting is given by

$$\mathscr{X} \sim \mathcal{N}_{n\times p}(\mathbf{M}, \mathbf{\Sigma}, \mathbf{\Psi}) \iff \text{vec}(\mathscr{X}) \sim \mathcal{N}_{np}(\text{vec}(\mathbf{M}), \mathbf{\Psi} \otimes \mathbf{\Sigma}),$$

where $\mathcal{N}_{np}(\cdot)$ is the np multivariate normal distribution, $\text{vec}(\cdot)$ is the vectorization operator, and $\otimes$ is the Kronecker product. A framework for assessing the matrix-variate normality of three-way data using both visual and goodness of fit tests is available in Pocuca *et al.* (2019).

2.2 Supervised, Unsupervised, and Semi-Supervised Learning

Machine learning algorithms can be broadly categorized into three varieties: supervised, unsupervised, and semi-supervised. The level of supervision indicates the presence and utilization of labeled data. In regression, labeled data corresponds to the observed values of the response variable associated with each input, while in classification, labels indicate group membership. Unsupervised learning is the most general case, where no observations are a priori labeled. In contrast, the other varieties

involve some labeled data, which is then used to infer labels for the unlabeled observations. In supervised learning, only the labeled data is used to infer labels for the unlabeled data. Semi-supervised learning leverages all available data to infer labels for the unlabeled data.

2.3 Model-Based Clustering

2.3.1 Historical Perspective

Classification is a paradigm in which group membership labels are assigned to unlabelled data. Unsupervised classification, or clustering, assumes that all observations are unlabelled or are treated as such. The groups to which observations are assigned are referred to as classes or clusters. Defining a cluster formally is surprisingly nontrivial. Intuitively, we want observations within a cluster to exhibit shared characteristics, making them more similar to each other than observations in other clusters. However, this definition is flawed because it elicits a solution where each observation is assigned to its own cluster McNicholas (2016a).

A more suitable definition can be constructed using finite mixture models. McNicholas (2016b) explains that the framing of a cluster in terms of a component of a mixture model can be traced back to Tiedeman (1955). Consider a population of G groups where the observations in each group are generated by a Gaussian density function. By censoring the group membership of each observation, we are left with a mixture of unknown densities. The reconstruction of the original G densities is known as clustering (Tiedeman, 1955). Subsequently, Wolfe (1963) considered a cluster to

be a component of a mixture model and pioneered multivariate Gaussian model-based clustering (Wolfe, 1965). Wolfe (1963) also explored two additional definitions of a cluster. The first definition characterizes a cluster as a mode in a distribution, while the second is based on the similarity between observations. McNicholas (2016a) synthesizes the historical development of a cluster and offers a refined definition as follows:

> A cluster is a unimodal component within an appropriate finite mixture model.

In this context, an appropriate finite mixture model is one that exhibits the necessary flexibility and parameterization to effectively fit the data.

2.3.2 Finite Mixture Models

The main objective of model-based clustering is maximize the likelihood of a G-component finite mixture model. While model-based clustering is typically applied to two-way data, it can be naturally extended to accommodate three-way data using matrix-variate distributions. A matrix-variate random variable $\mathscr{X}$ arises from a finite mixture model if its density can be written as

$$f(\mathbf{X}|\boldsymbol{\vartheta}) = \sum_{g=1}^{G} \pi_g f_g(\mathbf{X}|\boldsymbol{\theta}_g),$$

where $f_g(\cdot)$ is the gth component density, $\pi_g > 0$ is the gth mixing proportion with $\sum_{g=1}^{G} \pi_g = 1$, and $\boldsymbol{\vartheta} = (\boldsymbol{\pi}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_G)$ is the vector of parameters with $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_G)$. In most applications, the component density functions are taken to

be identical conditioned on $\boldsymbol{\theta}_g$, indicated as

$$f_g(\mathbf{X}|\boldsymbol{\theta}_g) = f(\mathbf{X}|\boldsymbol{\theta}_g),$$

for all g. Building on the ideas in Wolfe (1965), Viroli (2011) introduces the use of finite mixtures of matrix-variate normals for clustering three-way data. In the subsequent sections, we denote the density of a mixture of matrix-variate normals as

$$f(\mathbf{X}|\boldsymbol{\vartheta}) = \sum_{g=1}^{G} \pi_g \phi(\mathbf{X}|\mathbf{M}_g, \boldsymbol{\Sigma}_g, \boldsymbol{\Psi}_g),$$

where $\boldsymbol{\vartheta} = (\pi_1, \ldots, \pi_g, \mathbf{M}_1, \boldsymbol{\Sigma}_1, \boldsymbol{\Psi}_1, \ldots, \mathbf{M}_G, \boldsymbol{\Sigma}_G, \boldsymbol{\Psi}_G)$.

2.3.3 Parameter Estimation

Estimation of the mixture model parameters is typically performed using the expectation-maximization (EM) algorithm (Dempster *et al.*, 1977). The EM algorithm is an iterative procedure for finding maximum likelihood estimates under incomplete data. Consider a dataset comprising N unlabelled matrices $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ of size $n \times p$. Then the observed-data likelihood for a finite matrix-variate normal mixture is given by

$$\mathcal{L}(\boldsymbol{\vartheta}) = \prod_{i=1}^{N} \sum_{g=1}^{G} \pi_g \phi(\mathbf{X}_i|\mathbf{M}_g, \boldsymbol{\Sigma}_g, \boldsymbol{\Psi}_g).$$

Taking the natural logarithm gives us the observed-data log-likelihood

$$l(\boldsymbol{\vartheta}) = \sum_{i=1}^{N} \log \left(\sum_{g=1}^{G} \pi_g \phi(\mathbf{X}_i|\mathbf{M}_g, \boldsymbol{\Sigma}_g, \boldsymbol{\Psi}_g) \right).$$

To complete the data, we introduce latent membership indicators z_{ig} where

$$z_{ig} = \begin{cases} 1 & \text{if observation } \mathbf{X}_i \text{ belongs to component } g, \\ 0 & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, N; g = 1, \ldots, G$. Under this terminology, the goal of model-based clustering is to accurately predict z_{ig} for each observation and each component. Next, the indicators are organized into row vectors $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$ where $\mathbf{z}_i = (z_{i1}, z_{i2}, \ldots, z_{iG})$ denotes the membership label for data point i. The collection of all membership labels can be further condensed into an $N \times G$ solution matrix denoting a clustering of the data

$$\boldsymbol{\mathcal{Z}} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_N \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1G} \\ z_{21} & z_{22} & \cdots & z_{2G} \\ \vdots & \vdots & \ddots & \vdots \\ z_{N1} & z_{N2} & \cdots & z_{NG} \end{bmatrix}.$$

Using the data and membership labels, we obtain the complete-data likelihood and complete-data log-likelihood as follows

$$\mathcal{L}_c(\boldsymbol{\vartheta}) = \prod_{i=1}^{N} \prod_{g=1}^{G} \left[\pi_g \phi(\mathbf{X}_i | \mathbf{M}_g, \boldsymbol{\Sigma}_g, \boldsymbol{\Psi}_g)\right]^{z_{ig}},$$

$$l_c(\boldsymbol{\vartheta}) = \sum_{i=1}^{N} \sum_{g=1}^{G} z_{ig} \left[\log \pi_g + \log \phi(\mathbf{X}_i | \mathbf{M}_g, \boldsymbol{\Sigma}_g, \boldsymbol{\Psi}_g)\right].$$

For each iteration of the EM algorithm, the E-step replaces the z_{ig} indicators with their conditional expected values given the data and the current parameter estimates

$$\hat{z}_{ig} = \mathbb{E}[z_{ig} | \mathbf{X}_i, \hat{\boldsymbol{\vartheta}}] = \mathbb{P}[z_{ig} = 1 | \mathbf{X}_i, \hat{\boldsymbol{\vartheta}}] = \frac{\hat{\pi}_g \phi(\hat{\mathbf{X}}_i | \hat{\mathbf{M}}_g, \hat{\boldsymbol{\Sigma}}_g, \hat{\boldsymbol{\Psi}}_g)}{\sum_{h=1}^{G} \hat{\pi}_h \phi(\hat{\mathbf{X}}_i | \hat{\mathbf{M}}_h, \hat{\boldsymbol{\Sigma}}_h, \hat{\boldsymbol{\Psi}}_h)}.$$

The predictions can be reported as either soft or hard classifications. In soft classification, the values of $\hat{z}_{ig}$ remain as probabilities in $(0, 1)$, as computed during the E-step. Alternatively, hard classification reports the maximum a posteriori (MAP) classification i.e, MAP$\{\hat{z}_{ig}\}$, where

$$\text{MAP}\{\hat{z}_{ig}\} = \begin{cases} 1 & \text{if } g = \text{argmax}_h\{\hat{z}_{ih}\}, \\ 0 & \text{otherwise.} \end{cases}$$

In the M-step, the parameters are updated. Viroli (2011) derives the matrix-variate normal maximum likelihood estimates as

$$\begin{aligned} \hat{\pi}_g &= \frac{N_g}{N}, \quad N_g = \sum_{i=1}^{N} \hat{z}_{ig}, \\ \hat{\mathbf{M}}_g &= \frac{\sum_{i=1}^{N} \hat{z}_{ig}\mathbf{X}_i}{N_g}, \\ \hat{\mathbf{\Sigma}}_g &= \frac{\sum_{i=1}^{N} \hat{z}_{ig}(\mathbf{X}_i - \hat{\mathbf{M}}_g)\hat{\mathbf{\Psi}}_g^{-1}(\mathbf{X}_i - \hat{\mathbf{M}}_g)'}{pN_g}, \\ \hat{\mathbf{\Psi}}_g &= \frac{\sum_{i=1}^{N} \hat{z}_{ig}(\mathbf{X}_i - \hat{\mathbf{M}}_g)\hat{\mathbf{\Sigma}}_g^{-1}(\mathbf{X}_i - \hat{\mathbf{M}}_g)'}{nN_g}, \end{aligned}$$

for $g = 1, \ldots, G$. The E and M steps iterate until some convergence criterion is satisfied.

2.3.4 Convergence Criterion

To determine convergence of the EM algorithm, we use a criterion based on the Aitken acceleration (Aitken, 1926) defined by

$$a^{(t)} = \frac{l^{(t+1)} - l^{(t)}}{l^{(t)} - l^{(t-1)}},$$

where $l^{(t)}$ is the observed log-likelihood at iteration t. From this, Lindsay (1995) and Böhning *et al.* (1994) calculate the asymptotic estimate of the log-likelihood at iteration $t+1$ as

$$l_{\infty}^{(t+1)} = l^{(t)} + \frac{1}{1-a^{(t)}}(l^{(t+1)} - l^{(t)}).$$

In accordance with McNicholas *et al.* (2010), the EM algorithm terminates when

$$0 < l_{\infty}^{(t+1)} - l^{(t)} < \varepsilon,$$

for some pre-specified tolerance $\varepsilon > 0$.

2.3.5 Model Selection

In general, the number of underlying components or clusters is a priori unknown. To select the appropriate number, we use the Bayesian information criterion (BIC; Schwarz, 1978) defined as

$$\text{BIC} = 2l(\hat{\boldsymbol{\vartheta}}) - 2\rho \log N,$$

where $l(\hat{\boldsymbol{\vartheta}})$ is the maximized log-likelihood, N is the number of observations, and ρ is the number of free parameters. The model with the largest BIC value is considered the most appropriate. The use of the BIC for mixture-models has theoretical support in Leroux (1992) and Keribin (2000) where under certain regularity conditions, the correct number of components is consistently estimated. Dasgupta and Raftery (1998) also demonstrates BIC's value in selecting the appropriate number of components for multivariate Gaussian mixture models.

2.3.6 Model Performance

In the subsequent clustering, labeled datasets are treated as if they were unlabelled. Accordingly, we can assess model performance by comparing MAP classifications to the ground truth. The Rand index (RI; Rand, 1971) serves as a metric for quantifying the agreement between two sets of class assignments and is defined by

$$\mathrm{RI} = \frac{\text{number of pairwise agreements}}{\text{total number of pairs}}.$$

Pairwise agreement occurs when two observations are correctly assigned labels in relation to each other. If they belong to the same cluster, they should be labeled identically, while if they belong to different clusters, they should receive distinct labels. Conversely, a pairwise disagreement indicates inconsistent labels assigned to two observations. Together, pairwise agreements and disagreements encompass all possible pairs. The RI ranges between 0 and 1, with a value of 1 indicating a perfect match in the assigned classes. Under random assignment, the RI has positive expected value. To address this, Hubert and Arabie (1985) proposed the adjusted Rand index (ARI), which scales the RI to correct for the expected number of random pairwise agreements. The ARI is defined as

$$\mathrm{ARI} = \frac{\mathrm{RI} - \mathbb{E}[\mathrm{RI}]}{\max(\mathrm{RI}) - \mathbb{E}[\mathrm{RI}]}.$$

Consequently, the ARI has an expected value of 0 under random assignment and achieves a value of 1 for perfect assignment. It is important to note that the ARI permits negative values, indicating worse than random assignment.

2.3.7 Benefits Over Vectorization

A common approach to tackling complex problems is to transform them into a domain that is better understood. When dealing with matrix-variate clustering, it is tempting to revert to the multivariate setting through vectorization, permitting the use of well-established techniques. However, matrix-variate analysis often offers a substantial reduction in the number of free scale parameters. Gallaugher and McNicholas (2018) demonstrates a free scale parameter reduction from $(n^2p^2 + np)/2$ to $(n^2 + p^2 + n + p)/2$, which simplifies and speeds up the procedure for almost all values of n and p. Therefore, it is profitable to remain in the matrix-variate setting when dealing with three-way data.

2.4 Evolutionary Computation

Evolutionary computation (EC) is a paradigm inspired by the principles of biological evolution for global optimization. It explores the solution space without assuming any prior knowledge of its structure making it remarkably effective against challenging optimization problems. As an interdisciplinary enterprise, EC has many possible approaches. This treatment focuses on the perspective detailed in Ashlock (2010).

2.4.1 Evolutionary Algorithms

An evolutionary algorithm (EA) is a particular form of EC that exploits the concepts of reproduction and natural selection. The goal is to maximize a fitness function by evolving a population of candidate solutions. In contrast to other optimization techniques that yield only a single solution, EAs offer the advantage of obtaining a

population of optimal solutions.

To begin, the first generation of candidate solutions is initialized. For each generation, genetic operators such as crossover and mutation are applied to the population members giving rise to new individuals. Only the fittest individuals among the population survive, ensuring the propagation of advantageous traits. This process continues until the population stagnates to a collection of maximally fit solutions. In terms of natural selection, the traits of the final population have been selected due to their reproductive success. Note that the fitness of a particular solution is determined entirely by the fitness function under consideration. There can be several viable fitness functions, each potentially resulting in distinct optimal populations.

2.4.2 Genetic Operators

For reproduction, the principal mechanism is the genetic operator known as crossover. Crossover is the exchange of genetic material between two or more parent solutions to create at least one offspring. The genetic material is exchanged randomly at a single or multiple crossover points giving the offspring different trait combinations. This promotes exploration of the solution space by creating new and diverse candidate solutions. Over many generations, the offspring inherit favourable traits from their parents leading to fitter solutions. Mutation is a genetic operator that acts on individual population members by randomly introducing slight genetic modifications. This prevents premature convergence to suboptimal populations that may result from crossover alone.

The balance between these two operators is crucial for the efficacy of an EA. Natural processes indicate that crossover is to be the principal driving force of the

evolutionary process, while mutations are to be applied sparingly. Nonetheless, determining the precise ratio between these operators necessitates hyperparameter tuning to optimize the EA's performance.

2.4.3 Evolutionary Algorithms For Model-Based Clustering

Early applications of EAs for model-based clustering can be found in Martínez and Vitrià (2000) and Pernkopf and Bouchaffra (2005). To address the local maximum problem associated with the EM algorithm, the original works focused on evolving the parameters of a multivariate Gaussian mixture model. That is, they applied crossover and mutation directly to the parameters in $\boldsymbol{\vartheta}$. The fitness of potential solutions was assessed using the minimum description length (MDL).

More recently, Andrews and McNicholas (2013) and McNicholas *et al.* (2020) introduced a slightly different approach. Instead of evolving the parameter space, they applied crossover and mutation to a population of latent cluster membership matrices $\boldsymbol{\mathcal{Z}}_k$. In this case, the fitness function was the observed-log likelihood calculated at the parameter updates. The main advantage of evolving the cluster membership labels instead of the model parameters lies in the size of the respective spaces. The total number of possible classifications of N observations into G groups is finite, in contrast to the infinite parameter space (Andrews and McNicholas, 2013).

McNicholas *et al.* (2020) developed an EA for multivariate Gaussian model-based clustering, using two-point crossover and a greedy mutation function. In this work, we aim to extend this algorithm to the matrix-variate setting, allowing for broader applicability.

Chapter 3

Methods

The following sections introduce an EA for matrix-variate model-based clustering. First, the underlying model, population, and fitness function are described. Next is a presentation of the algorithm design, including a detailed discussion of the implementations of crossover and mutation. Finally, a procedure for applying the algorithm is provided.

3.1 Model and Fitness Function

The underlying model for our EA is a mixture of matrix-variate normal distributions. Representing individual clusterings of the data, matrices $\boldsymbol{\mathcal{Z}}_k$ consisting of indicator variables z_{ig} serve as population members. Unlike the EM algorithm which uses soft clustering, the EA estimates z_{ig} using hard clustering. For clarity, we denote $\hat{z}_{ig} \in [0, 1]$ for the EM estimates and $\tilde{z}_{ig} \in \{0, 1\}$ for the EA estimates. To compute the fitness of an individual solution, we first update the matrix-variate normal model

parameters, replacing $\hat{z}_{ig}$ with $\tilde{z}_{ig}$, that is,

$$\begin{aligned}
\tilde{\pi}_g &= \frac{N_g}{N}, \quad N_g = \sum_{i=1}^{N} \tilde{z}_{ig}, \\
\tilde{\mathbf{M}}_g &= \frac{\sum_{i=1}^{N} \tilde{z}_{ig}\mathbf{X}_i}{N_g}, \\
\tilde{\mathbf{\Sigma}}_g &= \frac{\sum_{i=1}^{N} \tilde{z}_{ig}(\mathbf{X}_i - \tilde{\mathbf{M}}_g)\tilde{\mathbf{\Psi}}_g^{-1}(\mathbf{X}_i - \tilde{\mathbf{M}}_g)'}{pN_g}, \\
\tilde{\mathbf{\Psi}}_g &= \frac{\sum_{i=1}^{N} \tilde{z}_{ig}(\mathbf{X}_i - \tilde{\mathbf{M}}_g)\tilde{\mathbf{\Sigma}}_g^{-1}(\mathbf{X}_i - \tilde{\mathbf{M}}_g)'}{nN_g},
\end{aligned}$$

for $g = 1, \ldots, G$. Then we calculate the observed log-likelihood at the current EA estimates

$$\text{fitness}(\tilde{\boldsymbol{\mathcal{Z}}}_k) = \sum_{i=1}^{N} \log\left(\sum_{g=1}^{G} \pi_g f(\mathbf{X}_i|\tilde{\mathbf{M}}_g, \tilde{\mathbf{\Sigma}}_g, \tilde{\mathbf{\Psi}}_g)\right).$$

3.2 Algorithm Design

The EA starts by initializing the first generation with a population of K parent solutions, denoted as $\tilde{\boldsymbol{\mathcal{Z}}}_1, \ldots, \tilde{\boldsymbol{\mathcal{Z}}}_K$. In general, the parent solutions are randomly generated matrices representing random hard clusterings of the data. Alternatively, one can provide initial solutions, such as a hardened k-means or other handpicked solutions, to inform the evolutionary process.

Reproduction in the EA involves cloning the parents J times and performing crossover on each parent and clone. Crossover randomly exchanges two distinct rows between the parent solution and its clone creating an offspring with identical membership labels except for two observations. It's worth noting that this exchange can also be viewed as a single parent operation in which two distinct rows of a single parent are swapped. While this differs from the exploratory aspect of biological crossover,

it offers the advantage of monotonically increasing the fitness of the population. To illustrate this operation, suppose we randomly select labels $\tilde{\mathbf{z}}_2$ and $\tilde{\mathbf{z}}_N$ from the parent in Figure 3.1. Since the rows are distinct, we swap them to create offspring $\tilde{\boldsymbol{\mathcal{Z}}}_{k_j}$ with the same genetic material except for those two labels.

$$\tilde{\boldsymbol{\mathcal{Z}}}_k = \begin{bmatrix} \tilde{z}_{11} & \tilde{z}_{12} & \dots & \tilde{z}_{1G} \\ 0 & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \dots & 0 \end{bmatrix} \xrightarrow{\text{crossover}} \tilde{\boldsymbol{\mathcal{Z}}}_{k_j} = \begin{bmatrix} \tilde{z}_{11} & \tilde{z}_{12} & \dots & \tilde{z}_{1G} \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Figure 3.1: Illustration of the crossover operator, randomly swapping distinct rows $\tilde{\mathbf{z}}_2$ and $\tilde{\mathbf{z}}_N$ from the parent solution to create a child with similar genetic material.

If the labels are identical, we continue randomly selecting rows until two distinct labels are found. After crossover, the population size grows to $K + KJ$, including the original parents and their offspring. The population is then organized into a list of descending fitness, from which we select the top K solutions to become the next generation of parents. This crossover step helps avoid stopping at local maxima of the fitness surface, i.e., the observed log-likelihood.

Because swapping alone does not guarantee better clustering results, we now introduce a greedy mutation step. For each of the surviving K individuals, we randomly swap the unit-element in a random row until their fitness improves. Suppose label $\tilde{\mathbf{z}}_2$ is randomly selected from the parent in Figure 3.2. Within this row, we may swap the unit-element $\tilde{z}_{2G} = 1$ with randomly selected zero-element $\tilde{z}_{22} = 0$.

$$\tilde{\boldsymbol{\mathcal{Z}}}_k = \begin{bmatrix} \tilde{z}_{11} & \tilde{z}_{12} & \dots & \tilde{z}_{1G} \\ \tilde{z}_{21} & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{z}_{N1} & \tilde{z}_{N2} & \dots & \tilde{z}_{NG} \end{bmatrix} \xrightarrow{\text{mutation}} \tilde{\boldsymbol{\mathcal{Z}}}_k = \begin{bmatrix} \tilde{z}_{11} & \tilde{z}_{12} & \dots & \tilde{z}_{1G} \\ \tilde{z}_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{z}_{N1} & \tilde{z}_{N2} & \dots & \tilde{z}_{NG} \end{bmatrix}$$

Figure 3.2: Illustration of the mutation operator, swapping unit-element $\tilde{z}_{2G}$ with zero-element $\tilde{z}_{22}$ in a random row of a surviving parent for a slightly modified solution.

If the mutation increases fitness, the swap remains; otherwise, the swap is reverted. This process continues until either a profitable mutation is found or all rows have been exhausted leaving the parent unchanged. If a generation remains unchanged after applying both crossover and mutation, it is considered a stagnation. After a predetermined number of consecutive stagnations, the algorithm terminates, resulting in a population of K fit solution matrices.

Algorithm 1 EA for matrix-variate model-based clustering.

Input:
$\boldsymbol{\mathcal{X}} = [\mathbf{X}_1, \ldots, \mathbf{X}_N] \leftarrow n \times p \times N$ array of observations
$\tilde{\mathfrak{Z}} = [\tilde{\boldsymbol{\mathcal{Z}}}_1, \ldots, \tilde{\boldsymbol{\mathcal{Z}}}_K] \leftarrow N \times G \times K$ array of membership labels ▷ random if not specified
$G \leftarrow$ number of clusters
$K \leftarrow$ number of parents
$J \leftarrow$ number of clones
$S \leftarrow$ max number of stagnations

```
s = 0
while s < S do
    for k = 1 to K do                                   ▷ crossover step
        for j = 1 to J do
            Crossover: randomly swap two distinct labels from parent Z̃_k to get offspring Z̃_kj
            Fitness: update model parameters and calculate log-likelihood of Z̃_kj
        end for
    end for
    Survival: sort parents and offspring by descending fitness and take top K as new parents
    for k = 1 to K do                                   ▷ mutation step
        for r in random permutation of 1 to N do
            Mutate: randomly swap unit-element with zero-element in row r
            if fitness increases then
                break for
            else
                revert the unit-element and zero-element swap
            end if
        end for
    end for
    if parents are identical to last generation then
        s ← s + 1
    else
        s ← 0
    end if
end while
Return final population Z̃
```

3.3 Procedure

The subsequent data analysis adopts the clustering paradigm, treating all membership labels as unknown. For each dataset, two matrix-variate normal mixtures are applied: one using the EM algorithm for parameter estimation and the other using our EA. Both algorithms are initialized with a random start. Let $\mathcal{G}$ be the true number of classes, then the algorithms are run for $G = 2, \ldots, \mathcal{G}+1$. The value of G which yields the largest BIC is selected. In addition to selecting G, the EA must also determine the number of parents $K \in \{1, 2, 3\}$, and the number of clones $J \in \{4, 8, 12\}$. The number of parents is relatively low to ensure that mutations are applied sparingly, while the number of children is large to encourage crossover. For convergence, the EM algorithm uses the Aitken's based criterion with a tolerance of $\varepsilon = 10^{-6}$, while the EA uses $S = 3$ stagnations. Model performance is based on the final converged log-likelihood, AIC, total runtime, and EA to EM likelihood ratio. The EM algorithm is implemented using the `R` package Tomarchio *et al.* (2023) and the EA is available in `Julia` at Flynn (2023) (R Core Team, 2023; Bezanson, Jeff and Edelman, Alan and Karpinski, Stefan and Shah, Viral B., 2023).

Chapter 4

Simulation

In the following sections, two simulations are conducted by clustering a collection of datasets generated from mixtures of matrix-variate normals. In each simulation, a total of 25 datasets are created from the same set of arbitrarily selected model parameters. Simulation 1 involves 3×4 data consisting of $\mathcal{G} = 2$ true classes and a total of $N = 300$ observations. The data is drawn to be balanced, with equal proportions in each class $\boldsymbol{\pi} = (\frac{1}{2}, \frac{1}{2})$. Simulation 2 considers 4×3 data generated from $\mathcal{G} = 3$ true classes, totalling $N = 300$ observations. Similarly, the data is balanced, with equal proportions in each class $\boldsymbol{\pi} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. The groups in Simulation 1 are well-separated, whereas the groups in Simulation 2 are more challenging to distinguish.

4.1 Simulation 1

In Simulation 1, the location parameters set to

$$\mathbf{M}_1 = \begin{bmatrix} 1 & 0 & 1 & -1 \\ -1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 1 & 0 & 1 & -1 \end{bmatrix},$$

and the scale parameters are given by

$$\mathbf{\Sigma}_1 = \begin{bmatrix} 1 & 0.4 & 0.75 \\ 0.4 & 1 & 0 \\ 0.75 & 0 & 1 \end{bmatrix}, \qquad \mathbf{\Sigma}_2 = \begin{bmatrix} 1 & 0.6 & 0.25 \\ 0.6 & 1 & 0.1 \\ 0.25 & 0.1 & 1 \end{bmatrix},$$

$$\mathbf{\Psi}_1 = \begin{bmatrix} 1 & 0 & 0.35 & 0.15 \\ 0 & 1 & 0 & 0.85 \\ 0.35 & 0 & 1 & 0 \\ 0.15 & 0.85 & 0 & 1 \end{bmatrix}, \qquad \mathbf{\Psi}_2 = \begin{bmatrix} 1 & 0.2 & 0 & 0.6 \\ 0.2 & 1 & 0.55 & 0 \\ 0 & 0.55 & 1 & 0.3 \\ 0.6 & 0 & 0.3 & 1 \end{bmatrix}.$$

The optimal number of components for the EM algorithm was $G = 2$ and the optimal EA consisted of $G = 2$, $K = 1$, and $J = 12$. Table 4.1 presents the mean and standard deviation for ARI, runtime, and EA to EM likelihood ratio of the optimal models across the 25 runs. The results indicate that both the EM and EA exhibited nearly identical performance. While the EA found slightly superior maxima, as evidenced by the average likelihood ratio, the EM achieved slightly better ARI scores. We acknowledge that the EA is generally slower than the EM; however, the runtimes appear comparable in this case since the number of parents is quite low.

Table 4.1: Mean and standard deviation of ARI, runtime, and likelihood ratio for EA and EM associated with Simulation 1.

	ARI	**Runtime (sec)**	**Likelihood Ratio**
EA	0.992 (0.010)	1.56 (0.19)	1.001 (0.005)
EM	0.993 (0.008)	0.77 (0.06)	

4.2 Simulation 2

The location parameters for simulation 2 are given by

$$\mathbf{M}_1 = \begin{bmatrix} 0 & 0.5 & 1 \\ 0.5 & 1 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} 1 & 0.5 & 0 \\ 1.5 & 1 & 2 \\ 0 & 2 & 0.5 \\ 1.5 & 0.5 & 1 \end{bmatrix}, \quad \mathbf{M}_3 = \begin{bmatrix} 1.5 & 2.5 & 2 \\ 1 & 3 & 1.5 \\ 0.5 & 3 & 1.5 \\ 1.5 & 0.5 & 1 \end{bmatrix},$$

and the scale parameters configured to

$$\mathbf{\Sigma}_1 = \mathbf{\Sigma}_3 = \begin{bmatrix} 1 & 0.1 & 0.45 & 0.1 \\ 0.1 & 1 & 0.25 & 0.35 \\ 0.45 & 0.25 & 1 & 0.1 \\ 0.1 & 0.35 & 0.1 & 1 \end{bmatrix}, \qquad \mathbf{\Sigma}_2 = \begin{bmatrix} 1 & 0.2 & 0 & 0.6 \\ 0.2 & 1 & 0.55 & 0 \\ 0 & 0.55 & 1 & 0.3 \\ 0.6 & 0 & 0.3 & 1 \end{bmatrix},$$

$$\mathbf{\Psi}_1 = \begin{bmatrix} 1 & 0.4 & 0.75 \\ 0.4 & 1 & 0 \\ 0.75 & 0 & 1 \end{bmatrix}, \qquad \mathbf{\Psi}_2 = \mathbf{\Psi}_3 = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0 \\ 0.5 & 0.5 & 1 \end{bmatrix}.$$

The number of components selected for the EM algorithm was $G = 3$, while the optimal EA was identified with $G = 3$, $K = 3$, and $J = 12$. Table 4.2 presents the mean and standard deviation for ARI, runtime, and EA to EM likelihood ratio of the

optimal models across the 25 runs. The results are similar to the previous simulation, with the EA exhibiting marginally superior performance in terms of likelihood, while the EM slightly outperformed in ARI scores. Notably, the larger number of parents resulted in significantly longer runtimes due to the greedy nature of the mutation step.

Table 4.2: Mean and standard deviation of ARI, runtime, and likelihood ratio for EA and EM associated with Simulation 2.

	ARI	**Runtime (sec)**	**Likelihood Ratio**
EA	0.930 (0.032)	14.72 (1.82)	1.041 (0.101)
EM	0.942 (0.026)	1.83 (0.42)	

Chapter 5

Application

The following sections detail an investigation of two real-world datasets. First is the Landsat satellite dataset, which involves 3×3 digital images of the same regions taken in four different spectral bands. The pixels are arranged into matrices of size 4×9 and three classes are used. The second dataset contains 16×16 greyscale images of handwritten digits. We focus on clustering digits 1 and 7.

5.1 Landsat Satellite Dataset

This dataset is a collection of digital images captured by the Landsat program, which is a series of Earth-observing satellites managed by NASA and the United States Geological Survey (USGS). The original data was purchased from NASA by the Australian Centre for Remote Sensing and subsequently donated to the UCI machine learning repository in a preprocessed form (Srinivasan, 1993). The preprocessing includes labelling the data based on site visits, removing data to prevent image reconstruction, and partitioning the data into training and testing sets.

Four digital images of size 3×3 were taken of the same region in different spectral bands, corresponding to random matrices of size 4×9. Two of the bands were in the visible spectrum, while the other two were in the infrared spectrum. The pixel intensities range from 0 to 255. The researchers identified seven classes of images based on the central pixels which are described in Table 5.1. Notably, class 6 was removed by the researchers due to doubts of the validity of the class.

Table 5.1: Description of the Landsat test set classes and observation count.

Class	**Description**	**Count**
1	Red soil	461
2	Cotton crop	224
3	Grey soil	396
4	Damp grey soil	211
5	Soil with vegetation stubble	237
6	Mixture of all classes	0
7	Very damp grey soil	470

Originally, this dataset was used for multivariate clustering by considering only a vector of the 4 central pixels, which is what the labels are based on. However, this approach neglects a significant amount of information along the borders which can be leveraged by matrix-variate distributions. Although we possess more information, the regions along the boundaries between classes are less distinct and pose a greater challenge to separate.

For our analysis, we focused on the first three classes from the test set: red soil, cotton crop, and grey soil, resulting in a dataset of size $N = 1081$ observations with true proportions approximately $\boldsymbol{\pi} = (0.43, 0.21, 0.37)$. Surprisingly, both the EM and EA resulted in $G = 4$ components. The presence of this extra class could be attributed to the lack of distinct borders between classes of images, as the images

were labeled based on their central pixels. In fact, it is possible that these bordering regions should have been assigned to class 6, which was later removed. The optimal EA also comprised $K = 2$ parents and $J = 8$ clones. Table 5.2 showcases the results, including the log-likelihood, ARI, total runtime, and the EA to EM likelihood ratio. Additionally, Tables 5.3 and 5.4 present the cross-tabulation of the MAP classifications for the EA and EM, respectively. The results demonstrate that, despite an increase of approximately 5 minutes in runtime, the EA discovered a superior maximum. Specifically, we have an EA to EM likelihood ratio of approximately 1.55 which translated to an approximate 0.01 increase in ARI. Notably, both algorithms exhibit a similar clustering pattern in the cross-tabulations.

Table 5.2: Converged log-likelihood, ARI, runtime, and EA to EM likelihood ratio associated with the Landsat dataset.

	Log-Likelihood	**ARI**	**Runtime (sec)**	**Likelihood Ratio**
EA	-108118.26	0.878	348.48	1.55
EM	-108118.7	0.869	36.89	

Table 5.3: Cross-tabulation of the EA MAP classifications associated with the Landsat dataset.

	Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**
Red Soil	452	0	0	9
Cotton Crop	0	140	0	84
Grey Soil	7	0	368	21

Table 5.4: Cross-tabulation of the EM MAP classifications associated with the Landsat dataset.

	Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**
Red Soil	450	0	0	11
Cotton Crop	0	150	0	74
Grey Soil	7	0	363	26

5.2 Handwritten Digits Dataset

The final dataset comprises greyscale images of handwritten digits, which were obtained by scanning envelopes from the U.S postal service. The original data is in binary format, and the images vary in size and orientation. To facilitate analysis, the Elements of Statistical Learning provides preprocessed training and test sets by centering, reorienting, resizing to 16×16, and normalizing the pixel intensities between -1 and 1 (Hastie *et al.*, 2009).

Since the writing is mostly concentrated in the center, the outer rows and columns primarily contain white space, resulting in a value of -1. Consequently, the lack of variation along the borders leads to singular updates in the scale matrices $\boldsymbol{\Sigma}_g$ and $\boldsymbol{\Psi}_g$. To address this sparsity, we introduced small random noise to the images. However, to preserve the signal integrity, a constant is applied to all entries greater than -1 before adding the noise.

Our analysis focused on clustering digits 1 and 7 from the dataset. There were a total of $N = 411$ observations from the test set, with 264 belonging to digit 1 and 147 to digit 7, corresponding to true proportions of approximately $\boldsymbol{\pi} = (0.64, 0.36)$. The optimal number of components for the EM algorithm was $G = 2$, while the optimal EA consisted of $G = 2, K = 2$, and $J = 4$. In Table 5.5, we present the log-likelihood, ARI, runtime, and EA to EM likelihood ratio. Additionally, Tables 5.6 and 5.7 display the cross-tabulation of EA and EM results, respectively.

Remarkably, both algorithms showed identical clustering results in terms of likelihood and ARI. The cross-tabulations for EA and EM were also identical. However, we note that the EA required significantly more runtime to achieve the same results as the EM algorithm.

Table 5.5: Converged log-likelihood, ARI, runtime, and EA to EM likelihood ratio associated with the handwritten digits dataset.

	Log-Likelihood	**ARI**	**Runtime (sec)**	**Likelihood Ratio**
EA	-97320.28	0.904	43.75	1.00
EM	-97320.28	0.904	2.95	

Table 5.6: Cross-tabulation of the EA MAP classifications associated with the handwritten digits dataset.

	Cluster 1	**Cluster 2**
Digit 1	256	8
Digit 7	2	145

Table 5.7: Cross-tabulation of the EM MAP classifications associated with the handwritten digits dataset.

	Cluster 1	**Cluster 2**
Digit 1	256	8
Digit 7	2	145

Chapter 6

Conclusions

6.1 Discussion

This work has extended the current literature on EAs for model-based clustering by incorporating mixtures of matrix-variate distributions. In particular, an EA was developed for matrix-variate normal mixtures using crossover and mutation. The EA was applied to simulated and real-world datasets and its performance was compared against the EM algorithm.

Simulation 1 featured datasets with well-separated groups generated by predetermined matrix-variate normal distributions. In Simulation 2, the datasets comprised additional groups that were more challenging to separate. Both the EA and EM algorithm performed similarly well on these datasets, with the EA showing a slightly superior likelihood performance and the EM algorithm having slightly better ARI performance.

Moving to real-world datasets, the Landsat satellite dataset presented a more difficult clustering problem due to the lack of distinct borders between images. Here, the

EA outperformed the EM algorithm in terms of both likelihood and ARI, showcasing its potential to handle complex and ill-defined problems more effectively. The second real-world dataset, containing handwritten digits of 1s and 7s, had a much larger dimension than the previous datasets. In this case, both the EA and EM performed equally well based on likelihood and ARI, although the EA struggled with runtime.

Overall, the EA proves to be a competitive alternative for fitting model parameters in matrix-variate normal mixtures, consistently performing at least as well as the EM algorithm in terms of likelihood and ARI. However, its limitation lies in its runtime, which is understandable given its approach of finding multiple solutions and the nature crossover and mutation operators. Considering the results obtained, it is recommended to deploy the EM algorithm in most cases, as it is computationally more efficient. The EA should be reserved for scenarios where the problem complexity is particularly high.

6.2 Future Work

Future research could focus on improving the EA's runtime performance, either by exploring more optimal implementations of the genetic operators, or by leveraging parallel computing. Additionally, extending the investigation to non-Gaussian distributions could broaden the applicability of the proposed method. Lastly, exploring modified versions of crossover and mutation could lead to further enhancements in the EA's performance.

Bibliography

Aitken, A. (1926). A series formula for the roots of algebraic and transcendental equations. *Proceedings of the Royal Society of Edinburgh*, **45**(1), 14–22.

Anderlucci, L. and Viroli, C. (2015). Covariance pattern mixture models for the analysis of multivariate heterogeneous longitudinal data. *The Annals of Applied Statistics*, **9**(2).

Andrews, J. L. and McNicholas, P. D. (2013). Using evolutionary algorithms for model-based clustering. *Pattern Recognition Letters*, **34**(9), 987–992.

Ashlock, D. (2010). *Evolutionary Computation for Modeling and Optimization*. Springer, New York.

Bezanson, Jeff and Edelman, Alan and Karpinski, Stefan and Shah, Viral B. (2023). The Julia Programming Language. Julia Computing.

Böhning, D., Dietz, E., Schaub, R., Schlattmann, P., and Lindsay, B. G. (1994). The distribution of the likelihood ratio for mixtures of densities from the one-parameter exponential family. *Annals of the Institute of Statistical Mathematics*, **46**(2), 373–388.

Dasgupta, A. and Raftery, A. E. (1998). Detecting features in spatial point processes with clutter via model-based clustering. *Journal of the American Statistical Association*, **93**(441), 294–302.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM-algorithm. *Journal of the Royal Statistical Society B*, **39**, 1–38.

Doğru, F., Bulut, Y., and Arslan, O. (2016). Finite mixtures of matrix variate t distributions. *Gazi Univ. J. Sci.*, **29**(2), 335–341.

Flynn, T. (2023). MatrixVariateEA: An Evolutionary Algorithm for Model-Based Clustering. `https://github.com/TommyJohnFlynn/MatrixVariateEA.jl`.

Gallaugher, M. P. B. and McNicholas, P. D. (2018). Finite mixtures of skewed matrix variate distributions. *Pattern Recognition*, **80**, 83–93.

Gupta, A. and Nagar, D. (1999). *Matrix Variate Distributions*. Monographs and Surveys in Pure and Applied Mathematics. Taylor & Francis.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, New York.

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

Keribin, C. (2000). Consistent estimation of the order of mixture models. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 49–66.

Leroux, B. G. (1992). Consistent estimation of a mixing distribution. *The Annals of Statistics*, pages 1350–1360.

Lindsay, B. (1995). Mixture models: Theory, geometry, and applications. In *NSF-CBMS Regional Conference Series in Probability and Statistics*, volume 5. Institute of Mathematical Statistics, Hayward.

Martínez, A. M. and Vitrià, J. (2000). Learning mixture models using a genetic version of the EM algorithm. *Pattern Recognition Letters*, **21**(8), 759–769.

McNicholas, P. D. (2016a). *Mixture Model-Based Classification*. Chapman & Hall/CRC Press, Boca Raton.

McNicholas, P. D. (2016b). Model-based clustering. *Journal of Classification*, **33**(3), 331–373.

McNicholas, P. D., Murphy, T. B., McDaid, A. F., and Frost, D. (2010). Serial and parallel implementations of model-based clustering via parsimonious Gaussian mixture models. *Computational Statistics & Data Analysis*, **54**(3), 711–723.

McNicholas, S. M., McNicholas, P. D., and Ashlock, D. A. (2020). An evolutionary algorithm with crossover and mutation for model-based clustering. *Journal of Classification*, pages 1–16.

Pernkopf, F. and Bouchaffra, D. (2005). Genetic-based EM algorithm for learning Gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(8), 1344–1348.

Pocuca, N., Gallaugher, M. P. B., Clark, K. M., and McNicholas, P. D. (2019). Assessing and visualizing matrix variate normality.

R Core Team (2023). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, **66**(336), 846–850.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, **6**(2), 461–464.

Silva, A., Qin, X., Rothstein, S. J., McNicholas, P. D., and Subedi, S. (2023). Finite mixtures of matrix variate Poisson-log normal distributions for three-way count data. *Bioinformatics*, **39**(5), btad167.

Srinivasan, A. (1993). Statlog (Landsat Satellite). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55887.

Tiedeman, D. V. (1955). On the study of types. In *Symposium on pattern analysis*.

Titterington, D., Smith, A., and Makov, U. (1985). *Statistical analysis of finite mixture distributions*. John Wiley & Sons, Chichester.

Tomarchio, S. D., Gallaugher, M. P., Punzo, A., and McNicholas, P. D. (2023). *MatrixMixtures: Model-Based Clustering via Matrix-Variate Mixture Models*. R package version 1.0.0.

Viroli, C. (2011). Finite mixtures of matrix normal distributions for classifying three-way data. *Statistical Computing*, **21**(4), 511–522.

Wolfe, J. (1963). *Object Cluster Analysis of Social Areas*. Master's thesis, University of California, Berkeley.

Wolfe, J. (1965). A computer program for the maximum-likelihood analysis of types. *USNPRA Technical Bulletin*, pages 15–65.