A Practical Introduction to Finite Mixture Models in R

Semester Paper

Wayne Zeng

Supervisor: Dr. Lukas Meier

Seminar for Statistics
D-MATH
ETH Zurich

Spring Semester 2022

Abstract

Finite mixture models are a flexible and convenient way to model unknown distributional shapes. They are also good at modelling group structures. A gentle introduction to mixture models will be given, as well as practical use cases in `R`, covering the package `mixtools`.

Contents

Introduction

Classification and subgrouping are important in many fields of data analysis. In many datasets, the memberships of each datapoint may not be easily discernible. The basic goal in such cases is to examine a sample of measurements and to differentiate these subgroups of individuals, even when there are no observable variables that readily indicate which subgroup an individual properly belongs. This leads to notions such as model-based clustering and unsupervised clustering.

Finite mixtures of distributions, a.k.a. *finite mixture models* or FMMs, are used to provide computationally convenient representations for modelling complex distributions of data on random phenomena. FMMs form the bases of cluster and latent class analyses, discriminant analysis and image analysis. They also provide descriptive models for distributions where a single component distribution is insufficient, i.e. they can give descriptions of entire subgroups, rather than assignments of individuals to those subgroups.

In Section 1, we introduce the definition of a finite mixture model. Section 2 explores latent variable models and the EM Algorithm as an alternative to maximum likelihood estimation. Section 3 then introduces the most commonly used FMM, the Gaussian Mixture Model, and lastly Section 4 applies the above to the Iris Dataset in `R`.

Much of Sections 1–3 closely follows McLachlan and Peel's "Finite Mixture Models" [4].

Acknowledgements

I would like to thank my supervisor, Lukas Meier, for his support and patience over the last few months. The fast, precise communication we were able to maintain was invaluable for my progress and understanding throughout the writing of this paper, and is greatly appreciated.

1 Formulation of Mixture Distributions

1.1 Basic Definition

Let $\mathbf{Y}_1, \dots, \mathbf{Y}_n$ denote a random sample of size $n \geq 1$, where $\mathbf{Y}_j$ is a p-dimensional random vector with probability density function (pdf) $f(\mathbf{y}_j)$ on $\mathbb{R}^p$. The vector $\mathbf{Y}_j$ contains the random variables corresponding to p measurements

Definition 1.1. The probability density function (pdf), or in the discrete case, the probability mass function, of a p-dimensional random vector $\mathbf{Y}$ takes the form

$$f(\mathbf{y}) = \sum_{i=1}^{g} \pi_i f_i(\mathbf{y}), \tag{1}$$

where the coefficients or "mixing proportions" π_i are non-negative and $\sum_{i=1}^{g} \pi_i = 1$, and $f_i(\mathbf{y})$ are the $g-$component densities, where $g \geq 1$ denotes the number of components in the mixture model.

The component densities may be parameterised with a vector $\boldsymbol{\theta}_i$ of parameters. We can then rewrite the above definition (1.1) as

$$f(\mathbf{y}; \boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i f_i(\mathbf{y}; \boldsymbol{\theta}_i), \tag{2}$$

where $\boldsymbol{\Psi} = (\pi_1, \pi_2, \dots, \pi_{g-1}, \boldsymbol{\xi}^T)$ is the vector of unknown parameters and $\boldsymbol{\xi}$ is the vector containing all the parameters in $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_g$ known in advance to be distinct.

1.2 Interpretation of Mixture Models

We can generate a random vector $\mathbf{Y}$ with g-component mixture density (1) by defining a new categorical random variable Z_j such that:

$$\begin{aligned} Z_j &= \{1, \dots, g\}, \\ P(Z_j = i) &= \pi_i, \\ P(\mathbf{Y}_j | Z_j = i) &= f(\mathbf{y}). \end{aligned}$$

Then the marginal density of $\mathbf{Y}$ is precisely $f(\mathbf{y})$.

We will see in Subsection 2.2.1 that it is convenient to work with a g-dimensional label.

1.3 Example: Mixtures of Two Normal Homoscedastic Components

Consider a mixture of two univariate normal components with common variance σ^2 and means μ_1, μ_2 with corresponding mixture proportions π_1, π_2 such that

$$f(y_i) = \pi_1 \phi(y_j; \mu_1, \sigma^2) + \pi_2 \phi(y_j; \mu_2, \sigma^2), \qquad \phi(y_j; \mu_i, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y_j - \mu_j)^2}{2\sigma^2}}. \tag{3}$$

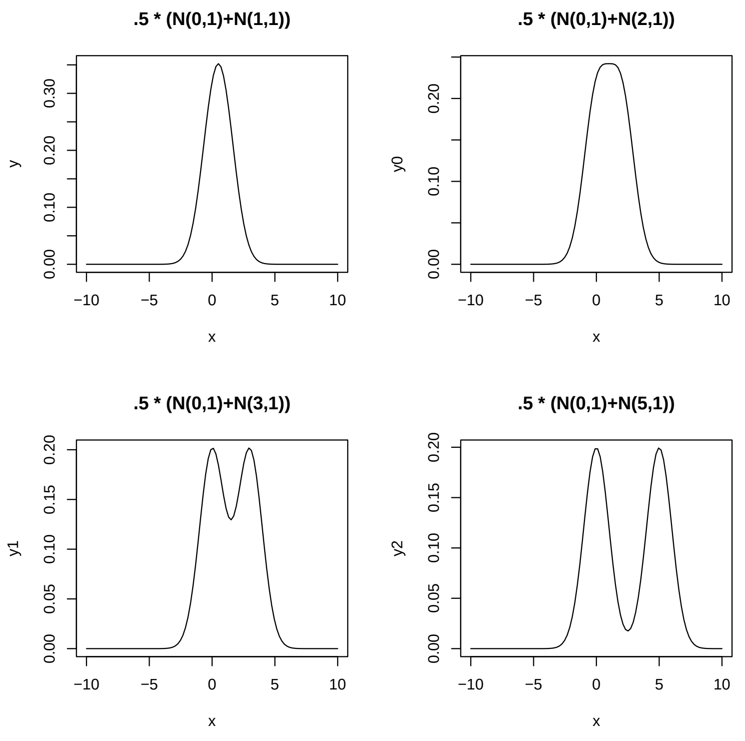
We consider two scenarios: one where the means of the two component normal densities are far apart vs. when they are sufficiently close together. If the two means are far apart, then we would expect (3) to resemble two normal densities side by side. For this purpose, let us define the Mahalanobis distance:

$$\Delta = |\mu_1 - \mu_2| / \sigma$$

In the other case where the means are more closely together, it is more difficult to see the distinction between them, leading to an asymmetric density if the components are not represented in equal proportions.

We now generate our first mixture model. Below is an example in R where we sum Gaussian components, each with means further away from each other than the last.

```r
set.seed(123)
par(mfrow=c(2,2))
x <- seq(-10, 10, length=100)
y <- .5 * (dnorm(x) + dnorm(x, mean=1))
y0 <- .5 * (dnorm(x) + dnorm(x, mean=2))
y1 <- .5 * (dnorm(x) + dnorm(x, mean=3))
y2 <- .5 * (dnorm(x) + dnorm(x, mean=5))
plot(x, y, type="l", main=".5 * (N(0,1)+N(1,1))")
plot(x, y0, type="l", main=".5 * (N(0,1)+N(2,1))")
plot(x, y1, type="l", main=".5 * (N(0,1)+N(3,1))")
plot(x, y2, type="l", main=".5 * (N(0,1)+N(5,1))")
```



1.4 Parametric Models

We will mostly be focusing on parametric mixture models. These models have the component densities $f_i(\mathbf{y}_j)$ belonging to some parametric family $\boldsymbol{\theta}_i$. We can then rewrite this as $f_i(\mathbf{y}_i; \boldsymbol{\theta}_i)$, $\boldsymbol{\theta}_i$ being the vector of unknown parameters in the postulated form for the ith component density in the mixture.

We rewrite (3) as

$$f(\mathbf{y}_j; \boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i f_i(\mathbf{y}_j; \boldsymbol{\theta}_j), \quad \boldsymbol{\Psi} = (\pi_1, \ldots, \pi_{g-1}, \boldsymbol{\xi}^T)^T$$

1.5 Identifiablty and Component Labels

Based on [5, Section 2.2].

For parametric families, $f(\mathbf{y};\boldsymbol{\Psi})$ is *identifiable* if the distinct values of $\boldsymbol{\Psi}$ determine distinct members of $\{f(\mathbf{y};\boldsymbol{\Psi}) : \boldsymbol{\Psi} \in \boldsymbol{\Omega}\}$, where $\boldsymbol{\Omega}$ is the parameter space. However, the mixture distribution definition is slightly different:

Definition 1.2. Let $f(\mathbf{y};\boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i f_i(\mathbf{y};\boldsymbol{\theta}_i)$ and $f(\mathbf{y};\boldsymbol{\Psi}^*) = \sum_{i=1}^{g^*} \pi_i^* f_i(\mathbf{y};\boldsymbol{\theta}_i^*)$ be two arbitrary members of a parametric family of mixture densities. Then this class of mixtures is *identifiable* for $\boldsymbol{\Psi} \in \boldsymbol{\Omega}$ if $f(\mathbf{y};\boldsymbol{\Psi}) \equiv f(\mathbf{y};\boldsymbol{\Psi}^*) \iff g = g^*$.

Intuitively, this means that we can permute the component labels so that $\pi_i = \pi_i^*$ and $f_i(\mathbf{y};\boldsymbol{\theta}_i) = f_i(\mathbf{y};\boldsymbol{\theta}_i^*)$, and that $f(\mathbf{y};\boldsymbol{\Psi})$ is invariant under the $g!$ permutations of said component labels.

In practice, a lack of identifiablity can be easily mitigated by, for example, imposing conditions on $\boldsymbol{\Psi}$ so that $\pi_1 \leq \pi_2 \leq \cdots \leq \pi_g$. However, issues then arise when we consider a Bayesian framework where posterior simulation is used to make inferences from the mixture model (also known as label-switching problem).

2 Maximum Likelihood Fitting of Mixture Models and the EM Algorithm

Loosely based on [2].

Maximum likelihood estimation (MLE) is the most commonly used approach to fitting mixture distributions and density estimation. This is done by searching across probability distributions and their parameters. However, MLE requires our dataset to be complete, since the likelihood no longer can be factorised and nor is it convex anymore, leading to a lack of a closed solution whilst maximising said likelihood. Furthermore, Bayesian inference of the parameters is made harder.

The *Expectation Maximisation Algorithm*, or EM, is used to perform MLE in the presence of latent variables. This is a general algorithm commonly used for density estimation with missing/latent data.

The EM algorithm is an iterative approach that cycles between two steps:

1. The first E-step (expectation) attempts to estimate the missing or latent variables.
2. The second M-step (maximisation) optimises the parameters of the model to explain the data.

We repeat this process until we reach convergence.

2.1 Latent Variable Models and Relationship to FMMs

In many statistical problems, there are many variables which cannot be measured directly. We denote these variables to be *latent*. We often link these latent variables to observable ones in order to infer the values of the latent ones.

Usually, statistical procedures and learning algorithms are used to estimate parameters of probability distributions to best fit the density of a given training dataset, mostly using maximum likelihood.

Let us consider the most simple case of a finite mixture model, where the dataset is comprised of many points that happen to be generated by two different processes. We generate the points to be Gaussian distributed, but the distributions are similar enough that it is not obvious to which distribution a given point may belong. The important remark here is that processes used to generate such data *represents* our latent variables. This clearly influences the data, but is not observable. As seen above, we cannot simply use MLE anymore, thus EM can be employed to estimate the parameters of a FMM.

2.2 Application of the EM Algorithm

2.2.1 Formulation as Incomplete Data Problem

We use the component label vector $\mathbf{Z}_j$ with the indicator variables. This notation is useful (although unintuitive at first) because it allows the MLE of the mixture distribution to be computed easily via the EM-algorithm.

We write

$$\mathbf{Y}_1, \ldots, \mathbf{Y}_n \overset{i.i.d.}{\sim} F,$$

where $F(\mathbf{y}_i)$ denotes the distribution function corresponding to the mixture density $f(\mathbf{y}_i)$.

The feature data $\mathbf{y}_1, \ldots, \mathbf{y}_n$ is incomplete since their associated component-indicator vectors $\mathbf{z}_1, \ldots, \mathbf{z}_n$ are not available, hence we consider this to be a latent variable model.

Furthermore, we define the complete data vector:

$$\mathbf{y}_c = (\mathbf{y}^T, \mathbf{z}^T)^T, \tag{4}$$

where

$$\mathbf{y} = (\mathbf{y}_1^T, \dots, \mathbf{y}_n^T)^T \quad \mathbf{z} = (\mathbf{z}_1^T, \dots, \mathbf{z}_n^T)^T$$

All the observations $\mathbf{y}_j$ have been completely recorded and fully observed. In addition, the component label vectors $\mathbf{z}$ are taken to be the realised values of the random vectors $\mathbf{Z}_1, \dots, \mathbf{Z}_n$

We can assume they are distributed unconditionally as

$$\mathbf{Z}_1, \dots, \mathbf{Z}_1, \overset{i.i.d.}{\sim} \text{Mult}_g(1, \boldsymbol{\pi}).$$

The ith mixing proportion π_i can be interpreted as the prior probability that the entity belongs to the ith component of the mixture distribution, whereas the posterior probability that the entity belongs to the ith component with $\mathbf{y}_j$ having been observed on it is given by

$$\begin{aligned}\tau(\mathbf{y}_j) &= P(\text{entity} \in i\text{th component} \mid \mathbf{y}_j) \\ &= P(Z_{ij} = 1 \mid \mathbf{y}_j) \\ &= \pi_i f_i(\mathbf{y}_j)/f(\mathbf{y}_j)\end{aligned}$$

This assumption means that the distribution of the complete data vector $\mathbf{Y}_c$ implies the appropriate distribution for the incomplete-data vector $\mathbf{Y}$. The final complete log likelihood for $\boldsymbol{\Psi}$ is

$$\log L_c(\boldsymbol{\Psi}) = \sum_{i=1}^{g} \sum_{j=1}^{n} z_{ij}\{\log \pi_i + \log f_i(\mathbf{y}_j; \boldsymbol{\theta}_i)\}. \tag{5}$$

2.2.2 EM Algorithm

Based on [4, Section 2.8].

Let us consider the standard FMM expression:

$$f(\mathbf{y}_j; \boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i f_i(\mathbf{y}_j; \boldsymbol{\theta}_i),$$

Where $\mathbf{y} = (\mathbf{y}_1^T, \dots, \mathbf{y}_n^T)^T$, $\boldsymbol{\Psi}$ is the observed incomplete data vector as defined above.

The log likelihood for $\boldsymbol{\Psi}$ formed *from the observed data is* given by the sum of the $f(\mathbf{y}_j; \boldsymbol{\Psi})$s:

$$\begin{aligned}\log L(\boldsymbol{\Psi}) &= \sum_{j=1}^{n} \log f(\mathbf{y}_j; \boldsymbol{\Psi}) \\ &= \sum_{j=1}^{n} \log\{\sum_{i=1}^{g} \pi_i f_i(\mathbf{y}_j; \boldsymbol{\theta}_i)\},\end{aligned}$$

and noting the difference between this expression and (5). To obtain the MLE, we would like to solve the likelihood equation

$$\frac{\partial \log L(\boldsymbol{\Psi})}{\partial \boldsymbol{\Psi}} = \mathbf{0}.$$

We denote the $\hat{\boldsymbol{\Psi}}$ as the MLE of $\boldsymbol{\Psi}$ and let

$$\tau_i(\mathbf{y}_i; \boldsymbol{\Psi}) = \frac{\pi_i f_i(\mathbf{y}_i; \boldsymbol{\theta}_i)}{\sum_{h=1}^{g} \pi_h f_h(\mathbf{y}_i; \boldsymbol{\theta}_h))} \tag{6}$$

be the posterior probability that $\mathbf{y}_j$ belongs to the ith component of the mixture. Then $\hat{\boldsymbol{\Psi}}$ satisfies the following expressions:

$$\hat{\pi_i} = \sum_{j=1}^{n} \tau_i(\mathbf{y}_i; \hat{\boldsymbol{\Psi}})/n \quad (i = 1, \ldots, g) \tag{7}$$

and

$$\sum_{i=1}^{g} \sum_{j=1}^{n} \tau_i(\mathbf{y}_i; \hat{\boldsymbol{\Psi}}) \partial \log_i(\mathbf{y}_i; \hat{\boldsymbol{\theta}}_i)/\partial \boldsymbol{\xi} = \mathbf{0}. \tag{8}$$

The two equations (7) and (8) build the iterative foundation of the EM algorithm. The proof of the two above equations involves Bayes' equation and can be found in [6, Section 1.4].

E-Step: Computing $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)})$

The E-step treats the latent data labels $\mathbf{z}_{ij}$s as missing. It handles the addition of these unobservable labels by taking the conditional expectation of the *complete* log-likelihood $\log L_c(\boldsymbol{\Psi})$ given $\mathbf{y}$ using the current fit for $\boldsymbol{\Psi}$. For the first step, we take this initial fit to be $\boldsymbol{\Psi}^{(0)}$. This initial value can be found by plugging into the right hand side of the equations.

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(0)}) = E_{\boldsymbol{\Psi}^{(0)}}\{\log L_c(\boldsymbol{\Psi}) | \mathbf{y}\}. \tag{9}$$

The subscript $\boldsymbol{\Psi}^{(0)}$ on the right hand side explicitly expresses that we are taking the expectation with respect to the current estimate $\boldsymbol{\Psi}^{(0)}$. In later iterations, we compute a new estimate for $\boldsymbol{\Psi}^{(k+1)}$ by taking the expectation w.r.t. $\boldsymbol{\Psi}^{(k)}$.

From (5), we see that $\log L_c(\boldsymbol{\Psi})$ is linear in the unobservable labels z_{ij}. This implies that

$$\begin{aligned} E_{\boldsymbol{\Psi}^{(k)}}\{Z_{ij} | \mathbf{y}\} &= \mathrm{pr}_{\boldsymbol{\Psi}^{(k)}}\{Z_{ij} = 1 | \mathbf{y}\} \\ &= \tau_i(\mathbf{y}_i; \boldsymbol{\Psi}^{(k)}) \\ &= \frac{\pi_i^{(k)} f_i(\mathbf{y}_i; \boldsymbol{\theta}_i^{(k)})}{f(\mathbf{y}_j; \boldsymbol{\Psi}^{(k)})} \\ &= \frac{\pi_i^{(k)} f_i(\mathbf{y}_i; \boldsymbol{\theta}_i^{(k)})}{\sum_{h=1}^{g} \pi_h^{(k)} f_h(\mathbf{y}_i; \boldsymbol{\theta}_h^{(k)}))} \end{aligned}$$

for $i = 1, \ldots, g$; $j = 1, \ldots, n$ in a similar fashion to (6).

Substituting this expression into (9), we obtain the final expression for the E-step:

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = \sum_{i=1}^{g} \sum_{j=1}^{n} \tau_i(\mathbf{y}_i; \boldsymbol{\Psi}^{(k)}) \{\log \pi_i + \log f_i(\mathbf{y}_j; \boldsymbol{\theta}_i)\}. \tag{10}$$

M-step: Maximising $Q(\mathbf{\Psi}; \mathbf{\Psi}^{(k)})$

The aim of the M-step is to globally maximise $Q(\mathbf{\Psi}; \mathbf{\Psi}^{(k)})$ w.r.t. $\mathbf{\Psi}$ over the parameter space Ω to produce the updated estimate $\mathbf{\Psi}^{(k+1)}$. Moreover, the mixture coefficients $\pi_i^{(k+1)}$ are calculated independently from $\boldsymbol{\xi}^{(k+1)}$.

Returning to the hypothetical case where the component labels z_{ij} would be observable. Then the MLE of π_i would be

$$\hat{\pi}_i = \sum_{j=1}^{n} z_{ij}/n. \tag{11}$$

In the E-step, we replaced each instance of z_{ij} with its current conditional expectation $\tau_i(\mathbf{y}_i; \mathbf{\Psi}^{(k)})$ in the complete data log-likelihood. In an analogous fashion, we can transform (11):

$$\pi_i^{(k+1)} = \sum_{j=1}^{n} \tau_i(\mathbf{y}_i; \mathbf{\Psi}^{(k)})/n. \tag{12}$$

This substitution is intuitively significant, since we see that each currently assessed posterior probability of membership $\pi_i^{(k+1)}$ contains a contribution from each observed component $\mathbf{y}_i$ in the mixture model. As for the parameter vector $\boldsymbol{\xi}$, we can take the partial derivative of the equation (10) and take one of the roots:

$$\sum_{i=1}^{g} \sum_{j=1}^{n} \tau_i(\mathbf{y}_i; \mathbf{\Psi}^{(k)}) \partial \log f_i(\mathbf{y}_j; \boldsymbol{\theta}_i)/\partial \boldsymbol{\xi} = 0. \tag{13}$$

This begs the question of whether the above equation has a closed form solution, since the original problem did not. In many cases, such as in a Gaussian mixture model, we do mostly have closed form solutions.

We cycle through the E-Step and M-step until we reach convergence, i.e.

$$L(\mathbf{\Psi}^{k+1}) - L(\mathbf{\Psi}^{k}) \leq \varepsilon, \ \varepsilon \text{ is usually set to } 1e-6$$

Furthermore, it was shown that the likelihood function is increasing after each EM iteration, i.e.

$$L(\mathbf{\Psi}^{k+1}) \geq L(\mathbf{\Psi}^{k}). \tag{14}$$

2.3 Starting Values and Initial Parameters

Based on[4, Sections 2.10, 2.12].

The EM algorithm requires an initial estimate of $\mathbf{\Psi}^{(0)}$. A good or bad initial estimate can affect the speed of convergence of the algorithm greatly. This is especially the case if the likelihood $L_c(\mathbf{\Psi})$ is unbounded on the edge of the parameter space Ω, potentially causing estimates for the $\mathbf{\Psi}^{(k)}$'s to be diverge if the initial value is chosen too closely to the aforementioned boundary. Further discussion of starting values can be found in [4, Section 3.8].

In addition, the likelihood equation (8) may possess multiple roots, prompting the use of multiple starting points in order to find all possible local maxima. In the case that there are no suitable roots, we choose the root with the largest MLE out of all the maxima.

A random approach may also be used to specifying initial values. In the Gaussian case with g normal components with means $\boldsymbol{\mu}_i$ and covariance matrices $\boldsymbol{\Sigma}_i$, we can randomly generate the starting means $\boldsymbol{\mu}_i^{(0)}$ as

$$\boldsymbol{\mu}_1^{(0)}, \ldots, \boldsymbol{\mu}_g^{(0)} \overset{i.i.d.}{\sim} N(\bar{\mathbf{y}}, \boldsymbol{V}),$$

where $\bar{\mathbf{y}}$ denotes the sample mean and $\boldsymbol{V}$ the sample covariance of the observed data. Using this expression, we can specify $\boldsymbol{\Sigma}_i$ and the mixing proportions π_i as

$$\boldsymbol{\Sigma}_i = \boldsymbol{V}, \qquad \pi_i^{(0)} = 1/g, \qquad i = 1, \ldots, g.$$

This method provides good variation between the initial values $\boldsymbol{\mu}_i^{(0)}$ and is not as computationally demanding as other random start methods.

However, in practice, basic clustering algorithms such as k-means can be employed as a way of finding starting points, as we will see in Section 4.

3 Gaussian and Other Mixture Models

This section is loosely based on [4, Sections 3.2 and 3.3].

The most common case of FMMs is where all of the mixture components are multivariate Gaussian. In the context of clustering, it is appropriate to use Gaussian mixture models (GMMs) whenever the clusters appear to have elliptical or symmetric structures. Any non-normal features in the data can then be easily identified and signal some other underlying group structure. Multivariate normal distributions are relatively simple and also make the M-step of the EM algorithm easier to compute. For clusters with heavier tails, we can also consider t-tailed mixture models.

3.1 Heteroscedastic Components

The original FMM equation (2) expressed with Gaussian mixture components takes the form

$$f(\mathbf{y}_j; \boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i \phi(\mathbf{y}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{15}$$

Note that $\boldsymbol{\xi}$ in the above case is comprised of the component means $\boldsymbol{\mu}_i$ and component covariance matrices $\boldsymbol{\Sigma}_i$, known in the heteroscedastic case to be distinct without restrictions.

The EM-Algorithm for GMMs is modified in the following way:

E-step: We modify $\tau_i(\mathbf{y}_i; \boldsymbol{\Psi})$

$$\tau_i(\mathbf{y}_i; \boldsymbol{\Psi}) = \frac{\pi_i \phi(\mathbf{y}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{h=1}^{g} \pi_h \phi(\mathbf{y}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}, \quad i = 1, \dots, g, \quad j = 1, \dots, n \tag{16}$$

M-step: the π_is are defined identically to the normal FMMs, see equation (12).

For $\boldsymbol{\mu}_i^{(k+1)}$ and $\boldsymbol{\Sigma}_i^{(k+1)}$ let us define the following notation:

$$\tau_{ij}^{(k)} = \tau_i(\mathbf{y}_j; \boldsymbol{\Psi}^{(k)}), \quad i = 1, \dots, g, \quad j = 1, \dots, n. \tag{17}$$

With this notation and using equation (13), replacing every f with ϕ, we obtain the following:

$$\boldsymbol{\mu}_i^{(k+1)} = \sum_{j=1}^{n} \tau_{ij}^{(k)} \mathbf{y}_j / \sum_{j=1}^{n} \tau_{ij}^{(k)}, \qquad \boldsymbol{\Sigma}_i^{(k+1)} = \sum_{j=1}^{n} \tau_{ij}^{(k)} (\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})(\mathbf{y}_j - \boldsymbol{\mu}_i^{(k+1)})^T / \sum_{j=1}^{n} \tau_{ij}^{(k)} \tag{18}$$

3.2 Homoscedastic Components

Now we would like to impose that all covariance matrices are identical, i.e.

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}$$

Then the M-Step simplifies down to

$$\boldsymbol{\Sigma}^{(k+1)} = \sum_{i=1}^{g} \left(\sum_{j=1}^{n} \tau_{ij}^{(k)} \right) \boldsymbol{\Sigma}_i^{(k+1)} / n. \tag{19}$$

The other coefficients π_i and $\boldsymbol{\mu}_i$ are identical to the heteroscedastic case.

3.3 Multivariate t-Mixtures

Based loosely on [5, Section 8.1] and [4, Section 7.1].

A majority of applied problems using FMMs have shorter, heavier tails of the normal distribution. We can employ t-mixtures, based on the t-distribution, in order to increase robustness and reduce the effect of outlying observations which are atypical of the components being fitted on the component means and covariance matrices, whereas a normal GMM would struggle.

The t-distribution has a new parameter ν called the number of *degrees of freedom* and has the following PDF:

$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})} \left(1 + \frac{t^2}{\nu}\right)^{-(\nu+1)/2}, \tag{20}$$

where Γ denotes the gamma function.

t-mixtures are a wider class of elliptically symmetric distributions with longer tails, making them more robust and less prone to extreme estimates. They are closely related to Gaussian mixtures; we can characterise (20) as being distributed $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i/u_j)$ given the realisation u_j of the latent random variable U_j; in addition, u_j and the component-indicator variables $\mathbf{z}_j$ in the EM framework are missing/latent. In fact, as $\nu \to \infty$, the t-distribution (20) goes to a normal distribution. Hence, ν can be treated as a robustness tuning variable.

A variant of the EM algorithm, called the *expectation-conditional maximisation* (ECM) algorithm, can be used for the MLE of multivariate t-components. ECM replaces the M-step of the traditional EM Algorithm by some simpler conditional maximisation steps. A more detailed explanation can be found in [4, Section 7.5].

4 Implementation with Examples in R: Iris Dataset

All code examples can be found here: [8]

There are multiple packages mentioned in [5] which are often used for finite mixture models and model-based clustering, such as `mixtools` [1], `EMMIX` (package written by the authors). Unfortunately, `EMMIX` is not available on Mac versions of `R`, so we will explore finite mixture models using `mixtools`.

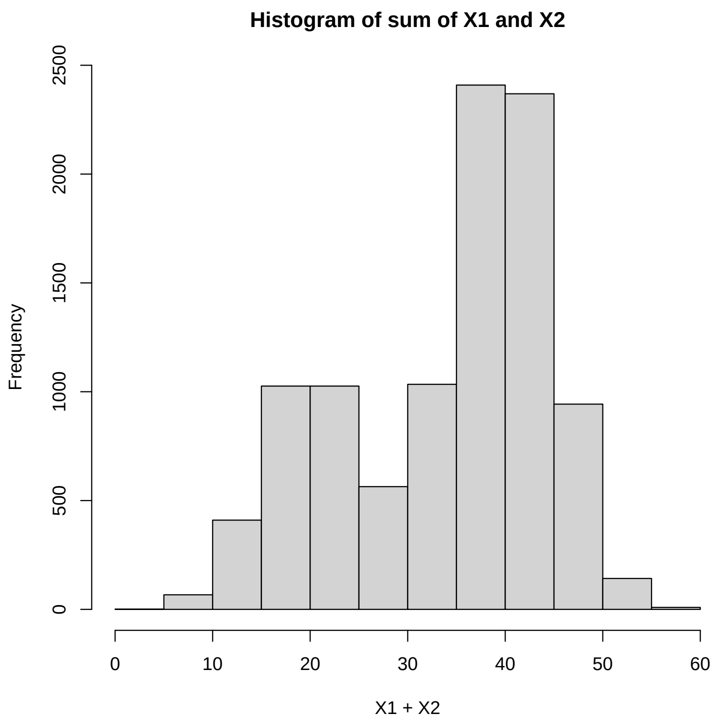
4.1 Synthetic Example by hand

This example is based on [9].

We begin with a synthetic example with two a sum of two normal Gaussians, firstly using `mixtools` to predict the coefficients $\pi_i, \boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$, and then we will go through the EM algorithm by hand.

We define two Gaussian distributions, one with 3000 samples from $N(20, 5)$ and the other with 7000 samples from $N(40, 5)$. We then concatenate both and plot the histogram:

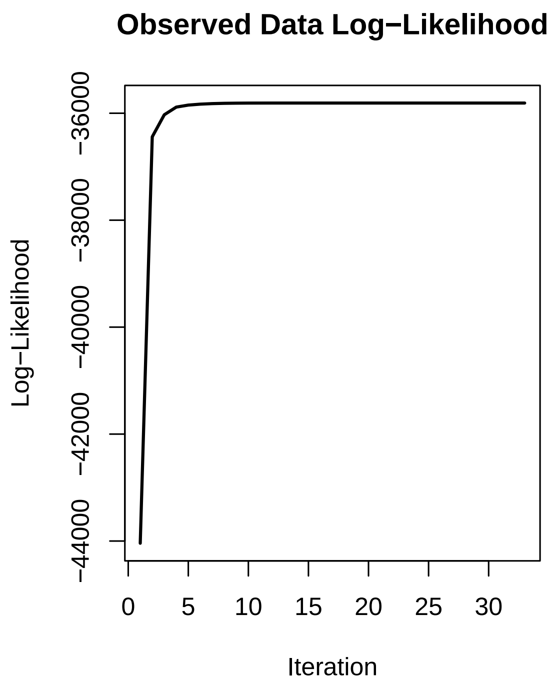
```
set.seed(123)
X1 = rnorm(3000, 20, 5)
X2 = rnorm(7000, 40, 5)
x = c(X1, X2)
hist(x, nclass=10, main="Histogram of sum of X1 and X2", xlab="X1 + X2")
```

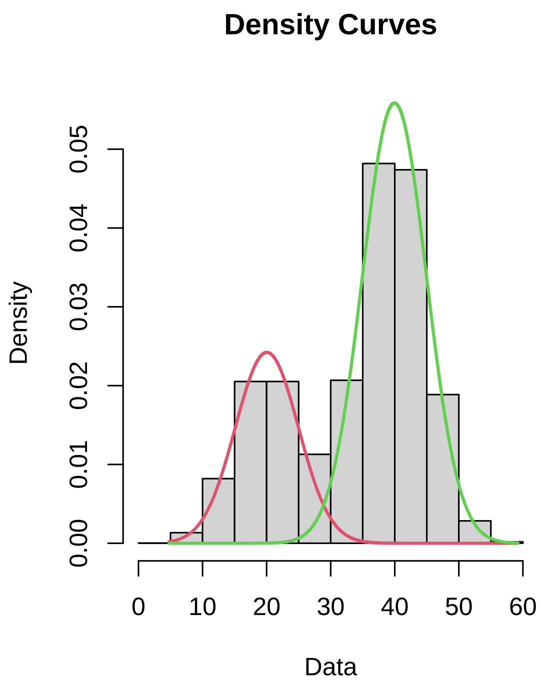


Using the `normalmixEM()` method from `mixtools`, we can plot the observed log-likelihood per iteration as well as the two density curves predicted:

```
suppressPackageStartupMessages(library(mixtools))
par(mfrow=c(1,2))
EMmixtool <- normalmixEM(x)

## number of iterations= 32
```

```
  plot(EMmixtool, density=TRUE)
```





We can then obtain the estimated values for $\pi_i, \boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$:

```
  EMmixtool[c("lambda", "mu", "sigma")]

## $lambda
## [1] 0.2994075 0.7005925
##
## $mu
## [1] 20.03593 39.95081
##
## $sigma
## [1] 4.932609 5.003639
```

Now we will execute the EM algorithm by hand and compare it to the values found by `normalmixEM()`. For this, we first inspect the histogram above which shows us two distinct peaks. This suggests that there are two major clusters and suggests using k-means with 2 clusters. We then store the means, standard deviations and mixture components:

```
mem <- kmeans(x,2)$cluster
mu1 <- mean(x[mem==1])
mu2 <- mean(x[mem==2])
sigma1 <- sd(x[mem==1])
sigma2 <- sd(x[mem==2])
pi1 <- sum(mem==1)/length(mem)
pi2 <- sum(mem==2)/length(mem)
```

We employ the method `sum.finite()` to omit any infinite values of x that may arise:

```
# modified sum only considers finite values
sum.finite <- function(x) {
  sum(x[is.finite(x)])
}
```

Subsequently, we calculate the initial value of $Q(\mathbf{\Psi};\mathbf{\Psi}^{(k)})$ by plugging into the right hand side of (9):

```
Q <- 0
# starting value of expected value of the log likelihood
Q[2] <- sum.finite(log(pi1)+log(dnorm(x, mu1, sigma1)))
  + sum.finite(log(pi2)+log(dnorm(x, mu2, sigma2)))

## [1] -95411.23
```

We closely follow the Gaussian version of the EM algorithm as in Section 3, paying attention to equations (16) for the E-step and (18) for the M-step:

```
k <- 2

while (abs(Q[k]-Q[k-1])>=1e-6) {
  # E step
  comp1 <- pi1 * dnorm(x, mu1, sigma1)
  comp2 <- pi2 * dnorm(x, mu2, sigma2)
  comp.sum <- comp1 + comp2

  p1 <- comp1/comp.sum
  p2 <- comp2/comp.sum

  # M step
  pi1 <- sum.finite(p1) / length(x)
  pi2 <- sum.finite(p2) / length(x)

  mu1 <- sum.finite(p1 * x) / sum.finite(p1)
  mu2 <- sum.finite(p2 * x) / sum.finite(p2)

  sigma1 <- sqrt(sum.finite(p1 * (x-mu1)^2) / sum.finite(p1))
  sigma2 <- sqrt(sum.finite(p2 * (x-mu2)^2) / sum.finite(p2))

  p1 <- pi1
  p2 <- pi2

  k <- k + 1
  Q[k] <- sum(log(comp.sum))
}
```

Comparing our results done "by hand" with the one from `normalmixEM`, our results align with the `mixtools` package:

```
  EMmixtool["lambda"]

## $lambda
## [1] 0.2994075 0.7005925

  c(pi2, pi1)

## [1] 0.2994099 0.7005901

  EMmixtool["mu"]

## $mu
## [1] 20.03593 39.95081

  c(mu2, mu1)

## [1] 20.03600 39.95084
```

```
  EMmixtool["sigma"]

## $sigma
## [1] 4.932609 5.003639

  c(sigma2, sigma1)

## [1] 4.932672 5.003607
```

4.2 Iris Dataset with GMMs

Based on the python code from [7].

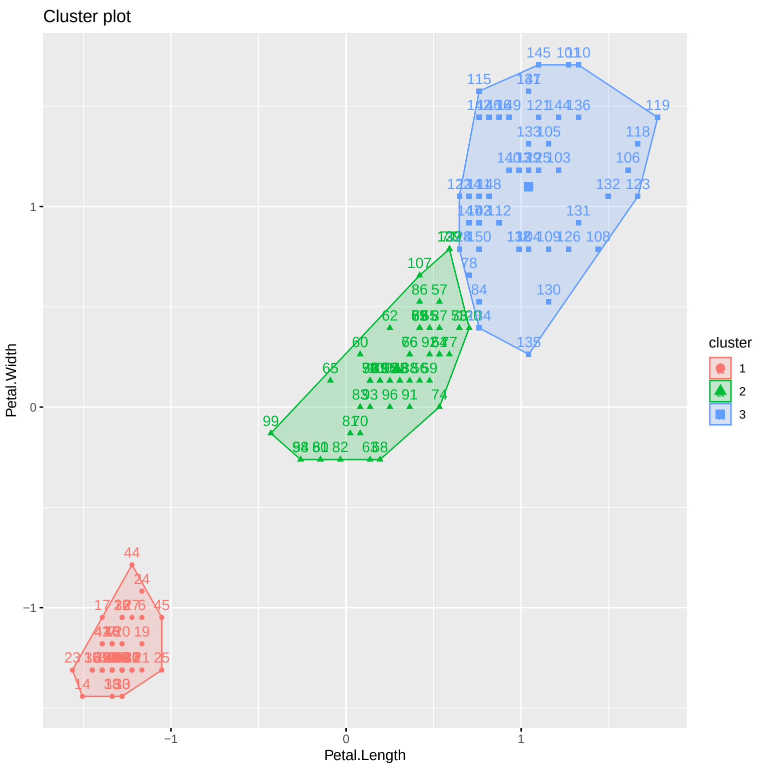
We now turn to a more practical example using the Iris dataset [3]. The data set consists of 50 samples from three species of the flower Iris. Four features: the lengths and widths of the sepals and petals were measured for each sample. It is one of the most commonly used datasets for basic statistical analysis.

We begin in a similar fashion to the previous example by importing `mixtools`, a new package `factoextra` as well as the Iris dataset itself.

```
  # Import packages and general graph setup
  suppressPackageStartupMessages(library(factoextra))
  suppressPackageStartupMessages(library(mixtools))
  data(iris)
  par(mfrow=c(1,2))
  set.seed(1234)
```

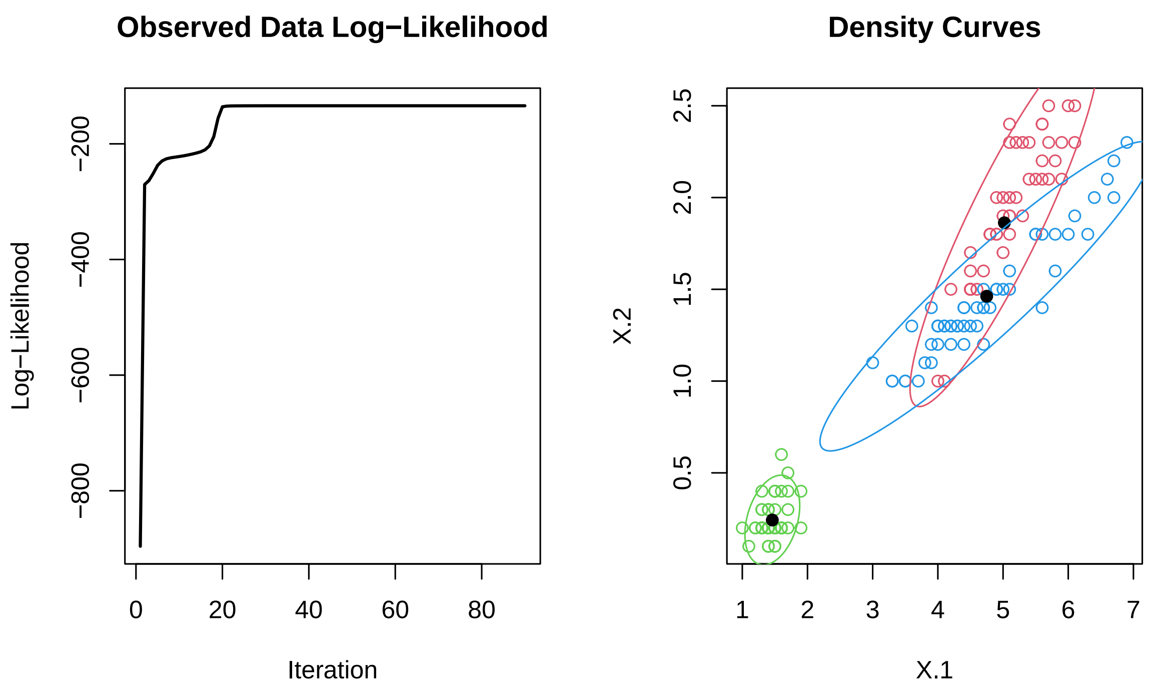
For this example, we will only consider the two features "Petal.Length" and "Petal.Width". We then perform k–means, knowing a priori that there will be three clusters involved. We use the method `fviz_cluster()` from `factoextra` to visualise the clusters at hand.

```
  iris.nolabel = iris[,3:4]
  k1 <- kmeans(x=iris.nolabel, centers=3)
  fviz_cluster(k1, data=iris.nolabel)
```



This time, however, running the multivariate normal EM algorithm, we run into a very inaccurate plot. The centers are classified far worse than k-means and each category overlaps each other extensively.

```
  irisEM <- mvnormalmixEM(iris.nolabel, k=3)

## number of iterations= 89

  suppressWarnings(summary(irisEM))

## summary of mvnormalmixEM object:
##           comp 1   comp 2
## lambda 0.348627 0.330089
## mu1    5.019893 1.862490
## mu2    1.460476 0.242997
## mu3    4.749200 1.462283
## loglik at estimate:  -134.1357

  par(mfrow=c(1,2))
  plot(irisEM, density=TRUE)
```

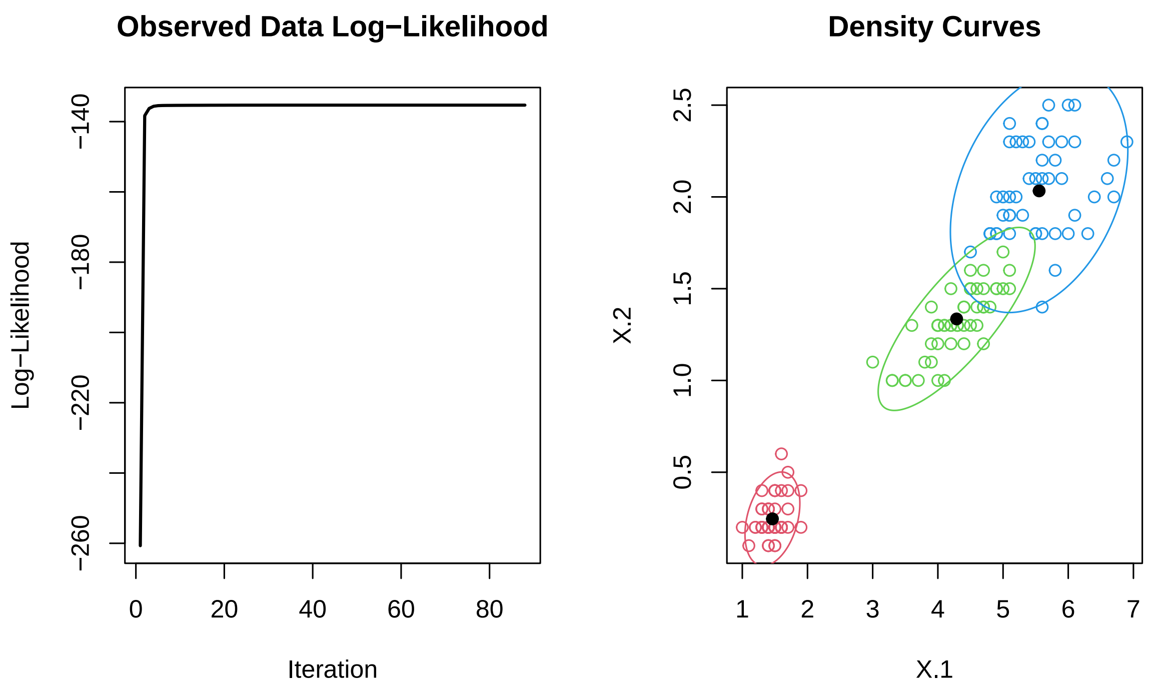


As mentioned in Subsection 2.3, this issue is likely cause by poor starting values. Using a similar approach to the first example, we would like to use k-means to then find better initial estimates for the parameter vector $\boldsymbol{\theta}_i$. This time, we must consider that our data is multivariate.

```
  mem <- k1$cluster

  pi1iris <- k1$size[1]/length(mem)
  pi2iris <- k1$size[2]/length(mem)
  pi3iris <- k1$size[3]/length(mem)

  mu1iris <- k1$centers[1,]
  mu2iris <- k1$centers[2,]
  mu3iris <- k1$centers[3,]
  sigma1iris <- diag(c(sd(iris.nolabel[mem==1,1]), sd(iris.nolabel[mem==1,2])))
  sigma2iris <- diag(c(sd(iris.nolabel[mem==2,1]), sd(iris.nolabel[mem==2,2])))
  sigma3iris <- diag(c(sd(iris.nolabel[mem==3,1]), sd(iris.nolabel[mem==3,2])))
```

Running `mvnormalmixEM()` again, we obtain a much improved estimation using the EM algorithm:

```
  irisEM2 <- mvnormalmixEM(iris.nolabel, lambda=c(pi1iris, pi2iris, pi3iris),
                         mu=list(mu1iris, mu2iris, mu3iris),
                         sigma=list(sigma1iris, sigma2iris, sigma3iris), k=3)

## number of iterations= 87

  par(mfrow=c(1,2))
  plot(irisEM2, density=TRUE)
```



The coefficient estimates are as follows:

```
  irisEM2$lambda

## [1] 0.3333329 0.3410062 0.3256608

  irisEM2$mu

## [[1]]
## [1] 1.4619996 0.2459998
##
## [[2]]
## [1] 4.287863 1.335230
##
## [[3]]
## [1] 5.553260 2.032826

  irisEM2$sigma

## [[1]]
##             [,1]       [,2]
## [1,] 0.02955588 0.00594794
## [2,] 0.00594794 0.01088395
##
## [[2]]
##             [,1]       [,2]
## [1,] 0.24167567 0.07951146
## [2,] 0.07951146 0.04148781
##
## [[3]]
##             [,1]       [,2]
## [1,] 0.30922741 0.05037616
## [2,] 0.05037616 0.07330311
```

References

[1] Tatiana Benaglia et al. “mixtools: An R Package for Analyzing Mixture Models”. In: *Journal of Statistical Software* 32.6 (2009), 1–29. DOI: `10.18637/jss.v032.i06`. URL: `https://www.jstatsoft.org/index.php/jss/article/view/v032i06`.

[2] Jason Brownlee. *A Gentle Introduction to Expectation-Maximization (EM Algorithm)*. 2019. URL: `https://machinelearningmastery.com/expectation-maximization-em-algorithm/` (visited on 09/20/2022).

[3] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[4] David Peel Geoffery J. McLachlan. *Finite Mixture Models*. Wiley, 2000.

[5] Suren I. Rathnayake Geoffery J. McLachlan Sharon X. Lee. “Finite Mixture Models”. In: *Annual Review of Statistics and Its Application* (2019). DOI: `https://doi.org/10.1146/annurev-statistics-031017-100325`.

[6] Thriyambakam Krishnan Geoffery J. McLachlan. *The EM Algorithm and Extensions*. Wiley, 1997.

[7] Dino Sejdinovic. *Example: Gaussian Mixtures on Iris Dataset*. 2019. URL: `https://www.stats.ox.ac.uk/~sejdinov/teaching/atsml19/Mixtures.html` (visited on 09/20/2022).

[8] *Wayne’s Github Repo for the Semester Project*. Accessed: 2022-09-20. URL: `https://github.com/way-ze/FMMs`.

[9] Haojun Zhu. *EM Algorithm Implementation*. 2016. URL: `https://rpubs.com/H_Zhu/246450` (visited on 09/20/2022).

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

A PRACTICAL INTRODUCTION TO FINITE MIXTURE MODELS IN R

Verfasst von (in Druckschrift):

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
ZENG	WAYNE SIXING

Ich bestätige mit meiner Unterschrift:
- Ich habe keine im Merkblatt „Zitier-Knigge“ beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Zürich, den 20. September 2022	[illegible]

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.